

APPLICATION OF THE DISCRETE ELEMENT METHOD TO STUDY THE
EFFECTS OF OCCLUSION INTERFACES IN SHALE

A Thesis

by

ANTU XIE

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Thomas A. Blasingame
Co-chair of Committee,	George J. Moridis
Committee Members,	Eduardo Gildin
	Zenon Medina-Cetina
Head of Department,	Dan Hill

December 2016

Major Subject: Petroleum Engineering

Copyright 2016 Antu Xie

ABSTRACT

The multiscale heterogeneity of ultratight shale rocks leads to the interesting yet pragmatic question of whether or not its microscale features may be used to predict macroscopic fracture behavior. Understanding the dominant parameters of microfracturing in these structures may help both with understanding the evolution of macroscopic fractures as well as permeability changes in shales. While many macroscopic analogs exist to correlate shale composition with fracture properties, few studies have examined the role that shale microstructure has on fracturing.

In this thesis, I first describe the method I developed to use SEM/EDS data from shale images to set up discrete element method simulations. I then explore the role of shale microstructures under standard uniaxial fracturing and what effect it may have on macroscopic material properties and if there is a special role that interfaces between different materials may play during fracturing in shales. Using the data provided and the simulation results, I demonstrate the qualitative role that the interfaces between different materials play during both compressive and tensile fracturing.

ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Blasingame, my co-chair, Dr. Moridis, and the members of my committee, Dr. Medina-Cetina, and Dr. Gildin. I would also like to thank my family who have always been there for me and provide me with infinite motivation to carry on.

CONTRIBUTORS AND FUNDING SOURCES

I thank the Crisman Institute and the Berg-Hughes Center for funding my exploration of microscale shale and applicability of DEM to exploring this topic.

TABLE OF CONTENTS

	Page
ABSTRACT	ii
ACKNOWLEDGMENTS	iii
CONTRIBUTORS AND FUNDING SOURCES	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	vii
LIST OF TABLES	xiii
CHAPTER I INTRODUCTION	1
1.1 Introduction and Problem Statement	1
1.2 Objectives	2
1.3 Results Summary	2
CHAPTER II LITERATURE REVIEW	4
2.1 Shale Definition and Macroscopic Properties	4
2.2 Shale Structure and Fracturing Relationships	6
2.3 Microscale Properties	7
2.4 Discrete Element Model and Implementation	11
CHAPTER III DEVELOPMENT AND IMPLEMENTATION OF METHODS	15
3.1 Motivation for Data Conversion and Simulation	15
3.2 General Data Conversion and Simulation Process	16
3.3 Material Properties	16
3.4 Mechanical Model	18
3.5 EDS Pre-Processing and Mapping Methodology	19
3.6 2D Particle Generation Methods	25
3.7 YADE-DEM Material Property Calibration	28
3.8 Effect of the Particles Count on Simulated Material Properties	30
3.9 Quantifying and Visualizing Fracture Events	34
CHAPTER IV UNIAXIAL FRACTURE STUDY	36
4.1 Uniaxial Test Overview	36
4.2 Unconfined Tensile Fracturing in Occlusion-Free Materials	39

4.3 Unconfined Tensile Fracturing in Shale Samples with Occlusions	46
4.4 Unconfined Compressive Fracturing	60
4.5 Discussion	70
CHAPTER V SUMMARY, CONCLUSION AND FUTURE WORK	71
5.1 Summary	71
5.2 Conclusions	72
5.3 Recommendations, Comments, and Future Work	72
REFERENCES	74
APPENDIX INSTRUCTIONS AND CODE TO RUN SIMULATION	79

LIST OF FIGURES

FIGURE	Page
2.1 Backscattered SEM images of various $4.8\ \mu\text{m} \times 4.1\ \mu\text{m}$ shales. (Curtis et al. 2012)	5
2.2 ESEM experiment showing the microcracks of a deformed and hydrated shale sample. Most microcracks are reported to occur in the fracture-matrix interface (Wang et al. 2016)	11
2.3 DEM models all materials as an assembly of elements. Two commonly seen element interactions are the normal and shear elastic interactions, controlled by a tensile modulus k_N and shear modulus k_s (Stránský, Jirásek, and Šmilauer 2010)	13
3.1 EDS digital output from converting a $82\ \mu\text{m} \times 56\ \mu\text{m}$ EDS image showing calcium on the left (black = No Calcium, red = Calcium) into the binary image on the right (Vitolini and Ajo-Franklin 2016, <i>Personal Communication</i>).	20
3.2 Rules that were implemented in order to define the material properties based on EDS information.	21
3.3 Full 2D material for Sample A. This map is generated from the rules outlined in Fig. 3.2 to help with assigning material properties for simulation purposes.	22
3.4 The final material map keeps track of both materials as well as whether or not a pixel is on the interface between two materials.	23
3.5 Material maps for all four samples. The dimensions of each sample are, respectively, $81\ \mu\text{m} \times 56\ \mu\text{m}$, $220\ \mu\text{m} \times 150\ \mu\text{m}$, $1400\ \mu\text{m} \times 990\ \mu\text{m}$, and $520\ \mu\text{m} \times 360\ \mu\text{m}$	24
3.6 Example of differences between gravity-settling particle growth packing. (A) the gravity growth model concentrates irregularity at the top but is more tightly packed (B) The growth model distributes irregularity throughout the model, but is more uniform at all boundaries.	26
3.7 Box and whisker plots showing differences in fracture stress and Young's Modulus estimates. 40 trials were run for each parameter in a simple $20\ \mu\text{m} \times 30\ \mu\text{m}$ uniaxial tension experiment for a homogeneous material. Grey and orange shows the first quartile	

	above and below the median, respectively, and whiskers show the absolute maximum and minimum Young's Modulus found. The growth method is slightly more accurate and takes far less time to generate an assembly of particles.	27
3.8	Image showing conversion of map to particle assembly. By using a material map such as the one for Sample A on the left, I can assign a material identity to every particle in the DEM simulation. In the image on the right, a 56,000 particle simulation is created by determining a particle's material properties from its position.	28
3.9	Box and whisker plot showing Macroscopic Young's Modulus vs. the number of particles. 20 trials were run for each parameter in a simple uniaxial tension experiment. Grey and orange shows the first quartile above and below the median, respectively, and whiskers show the absolute maximum and minimum Young's Modulus found. Increasing the number of particles only slightly increases calibration accuracy once the number of spheres approaches the original number of pixel.	32
3.10	Box and whisker plot showing fracture stress vs. the number of particles. 20 trials were run for each parameter in a simple uniaxial tension experiment. Grey and orange shows the first quartile above and below the median, respectively, and whiskers show the absolute maximum and minimum Young's Modulus found. Increasing the number of particles only slightly increases calibration accuracy once the number of spheres approaches the original number of pixel.	33
3.11	The material map for sample A. Each color represents a different mineral or compound.	35
4.1	During tensile testing, the edges of the DEM particle assembly is loaded in the horizontal direction. The tensile stress proceeds at a constant strain rate until a user-defined endpoint is encountered.	39
4.2	Particle assembly from all four samples, based on SEM/EDS images. As a first test, all non-empty spaces are converted to the dominant material of that sample. For samples A and B, the dominant material is clay. For samples C and D, the dominant materials are carbonate and quartz, respectively.	40
4.3	A simulated region based on the structure of sample A. Tensile loading is applied to maintain a strain rate of 100 m/s. Each of the other three DEM particle assemblies shown in Fig. 4 will have also	

have a tensile force applied upon them under a constant strain rate of 100 m/s.	41
4.4 Pure clay fracture patterns of sample A. The light blue section is defined as clay, the dark blue section is defined as pore space, and white dots represent bonds that have fractured during loading. The only difference between every trial is the packing of the particles themselves. Outlined in red is the fracture shape that is shared by all four representative samples.	42
4.5 Pure clay fracture patterns of sample B. In sample B, the light blue is clay, the dark blue section is pore space, and white dots represent bonds that fractured during tensile loading. The only difference between every trial is the packing of the particle assembly. In this case, because there are no overriding features in the mesh, no dominant fracture network has evolved yet.	43
4.6 Pure carbonate fracture patterns of sample C, the light green section is defined as carbonate and the white dots represent the position of bond fractures. While there is a general pattern to fracture vertically, I cannot observe predictable fracture location. Sample C contains no empty pores, so it is only the random packing of sample that results in these differences.	44
4.7 Pure quartz fracture patterns of sample D. The orange represents quartz, the dark blue is once again pore space, and white dots represent bonds that have fractured. Above, I can see four very similar fracture patterns, suggesting the dominant role that a single flaw can have on a structure.	45
4.8 Sample A DEM particle assembly, made from the material maps in Fig. 3.4	47
4.9 Sample B DEM particle assembly, made from the material maps in Fig. 3.4	47
4.10 Sample C DEM particle assembly, made from the material maps in Fig. 3.4	48
4.11 Sample D DEM particle assembly, made from the material maps in Fig. 3.4	48
4.12 Representative Unconfined Stress vs. Strain plot for sample A. The $Tan\delta$ method is used to determine modulus and represented by the	

blue dotted line. The region where a plastic regime is observed is shown by the red oval.	50
4.13 Representative Unconfined Stress vs. Strain plot for sample B. The $Tan\delta$ method is used to determine modulus and represented by the blue dotted line. The region where a plastic regime is observed is shown by the red oval.	51
4.14 Representative Unconfined Stress vs. Strain plot for sample C. The $Tan\delta$ method is used to determine Young's modulus and represented by the blue dotted line. Compared to Sample A and B, the modulus and fracture stress is significantly higher, likely because the dominant materials in this sample are carbonate and quartz, and also because there are no pores whatsoever in Sample C.	52
4.15 Representative Unconfined Stress vs. Strain plot for sample D. The $Tan\delta$ method is used to determine Young's modulus and represented by the blue dotted line. Compared to Sample A and B, the modulus and fracture stress is significantly higher, likely because the dominant materials in this sample are carbonate and quartz, and also because there are no pores whatsoever in Sample C.	53
4.16 Example of fractures that evolve under tension for Sample A. Unlike the fractures for a pure-clay material in Fig. 4.5, this fracture does not come in contact with the central pore because this pore is now surrounded by quartz occlusions that inhibit fracture growth.	56
4.17 Example of fractures that evolve under tension for Sample B. Unlike sample A, the distribution of mineral occlusions do not completely isolate any pore or significantly impede vertical fracture growth. Thus, these occlusions are less effective at strengthening the sample.	56
4.18 Example of fractures that evolve under tension for sample C. Without a significant quantity of clay, fracturing must occur within the minerals. A visual inspection of these fractures show that the majority virtually all occur at or near the boundary interfaces between quartz and carbonate.	57
4.19 Example of fractures that evolve under tension for sample D. Though there is a central pore, the quartz occlusion surrounding the pore on the bottom arrests fracture growth and induces fractures to grow at edges of the quartz occlusions instead.	57

4.20	Representative Confined Stress vs. Strain plot for sample A. In the first, uniaxial loading meets an elastic response from the material. However, enough stress is eventually applied that the material begins splitting across the central large pore (see Fig. 3.9 as a reminder of this pore's size relative to the geometry). Thus, even before reaching the critical fracture stress, this division results a weaker overall particle system.....	61
4.21	Representative Confined Stress vs. Strain plot for sample B. Sample B shows a stress-strain response to compression in a qualitatively similar way to Sample A. In the case of sample B, however, the arrangement of the pores and occlusion are much more accommodating to shear failure due to the distributed nature of the occlusions (see Fig. 4.10).	62
4.22	Representative Confined Stress vs. Strain plot for sample C. Sample C shows a significantly greater modulus due to its elevated percent volume of quartz and carbonate greater mineral count. We note that its estimated initial Young's modulus of elasticity moduli is actually less than the one estimated from tensile testing.....	63
4.23	Representative Confined Stress vs. Strain plot for sample D. As expected, sample D shows the greatest Young's modulus and fracture stress due to its relatively high quartz fraction. Though there are some pores in the volume, they amount to less than 6%.	64
4.24	Example of fractures that evolve under tension for sample A. As can be seen from Sample A under compression, compressive fractures are very different from tensile fractures. However, even in compression, fractures appear to be clustered around the interfaces of different materials and propagate from existing fractures.....	66
4.25	Example of fractures that evolve under tension for sample B. The general trend of fractures to occur at the interface between clay and mineral occlusion is also evident in the Sample B. One possible reason that the compressive stress limit is so much greater in Sample A is that, unlike the quartz in Sample A, the quartz occlusions in Sample B are relatively small and thus the clay mortar can continue to be primary mineral to be fractured without being 'trapped' by quartz.....	66
4.26	Example of fractures that evolve under tension for sample C. As can be seen from this image, the heavy quartz presence and lack of clay ensures that some degree of fracturing occurs in the carbonate component. Of course, this dramatically increases the stress needed to fracture the simulated material.	67

4.27	Example of fractures that evolve under tension for sample D. In Sample D, compression may induce fracture growth around the pore space, especially in carbonate. One may wonder why a similar effect is not noticed in Sample A (Fig. 4.24), though this may have to do with relative abundance of clay near the boundary which can already to deform.....	67
------	--	----

LIST OF TABLES

TABLE	Page
3.1 Mineral Material Properties Used to Calibrate DEM Bond Parameters	18
4.1 Tests to Be Applied to All 4 Samples	37
4.2 Simulation Parameters for Unconfined Loading	38
4.3 Material Properties Stresses (8 Trials per Sample)	46
4.4 Sample Composition	49
4.5 Young's Modulus and Fracture Stress under Tension (8 trials per sample)	54
4.6 Fracture-Interface Correlation at 2×10^{-5} strain (8 trials per sample)	59
4.7 Material Properties Estimate From Compression (8 Trials per Sample)	65
4.8 Fracture-Interface Correlation Under Compression (8 trials per sample)	69

CHAPTER I

INTRODUCTION

1.1 Introduction and Problem Statement

Traditional methods of simulating hydraulic fracture treatments focus on macroscopic mechanical phenomena as it relates to primary fractures. However, these models do not explore how such stresses may generate microfractures and further alter the properties of shales relevant to hydrocarbon production. While these fractures may individually have a negligible effect on reservoir properties, the cumulative effects of these cracks on the reservoir permeability may be significant. Additionally, their presence and evolution may affect the propagation, direction and other relevant characteristics of macroscopic fractures.

The potential parameters relevant to microscale fracturing in the subsurface are limited only by the number of rocks in the world, and researchers do not have infinite time to explore these parameters during their investigations. Fortunately, the nature of fracturing in heterogeneous materials suggests that it is often the points of highest stress concentration or weakest mechanical components that fracture, which suggests that there may be dominant parameters that allow us to neglect less important properties. In the case of shales with high-porosity or high-organic content, an understanding of the shale's pore structure may be sufficient to predicting mechanical properties. However, in the case of tight shales, these pores are likely to be distributed in insufficiently large concentration or are too small to completely dominate fracture phenomena. Thus, I will focus my investigation on an exploration of the role that specific features of the microscale structure

(i.e., occlusions of different materials) of shales may have on the initiation and propagation of microfractures. This investigation included developing a method to make use of existing microscale shale data, studying the role that shale micro-heterogeneity plays in shale fracturing, and exploring the role that interfaces between different materials may play in such interactions. Success in this effort can provide the impetus for further studies to quantify these properties. Moreover, an understanding of these parameters will allow for a better, physics-based approach towards estimating the permeability of rock near the macroscopic fracture walls after a fracture treatment.

1.2 Objectives

The objectives of this work are:

- To *develop* a method to study microscale shale fracturing phenomena that can make use of EDS/SEM image data.
- To *determine* if the microstructure of shales has any effect on macroscale or microscale fracturing properties, and if so, what microstructural properties are most important.

1.3 Results Summary

This study first documents a method of determining shale structure by combining data from scanning electron microscopy (SEM) and energy-dispersive spectroscopy (EDS), including outlining the edges (boundaries) of different materials. I then apply this method to four different samples of Niobrara shale and calibrate the model using material

properties obtained from literature. Finally, I study by means of numerical simulation the mechanical behavior of the four samples of shale under conditions of uniaxial tensile stress and compressive stress in order to obtain qualitative insights into fracture phenomena. The results of this study qualitatively support the thesis that the interface between different components of a heterogeneous material is the feature most susceptible to fracturing, and thus may strongly influence fracture initiation and propagation.

CHAPTER II

LITERATURE REVIEW

2.1 Shale Definition and Macroscopic Properties

As a term, 'shale' describes a wide variety of clay structures, with alternative names such as mudrock, claystone, and argillaceous material. For the purposes of this thesis, I will accept the definition of shale as "characterized by fine grain and lamination... any rock type containing at least 30% clay will be known as engineering shales" (Asef and Farrokhrouz 2013). Such a loose definition allows a multitude of materials with any number of additional components such as quartz, kerogen, or pyrite to be classified as shale. In addition to any number of possible compositions, the various microstructures and nanostructures possible all contribute even greater variability to the bulk material properties of shale. For example, Curtis et al. (2012) provided backscattered SEM images of various $4.8\ \mu\text{m} \times 4.1\ \mu\text{m}$ shales (**Fig. 2.1**), revealing different compositions and microstructures. The variety of greyscale brightness in each image shows that shales can be quite diverse when observing their microstructure and nanostructure, and that assuming they all have the same structure between formations may result in significant inaccuracies.

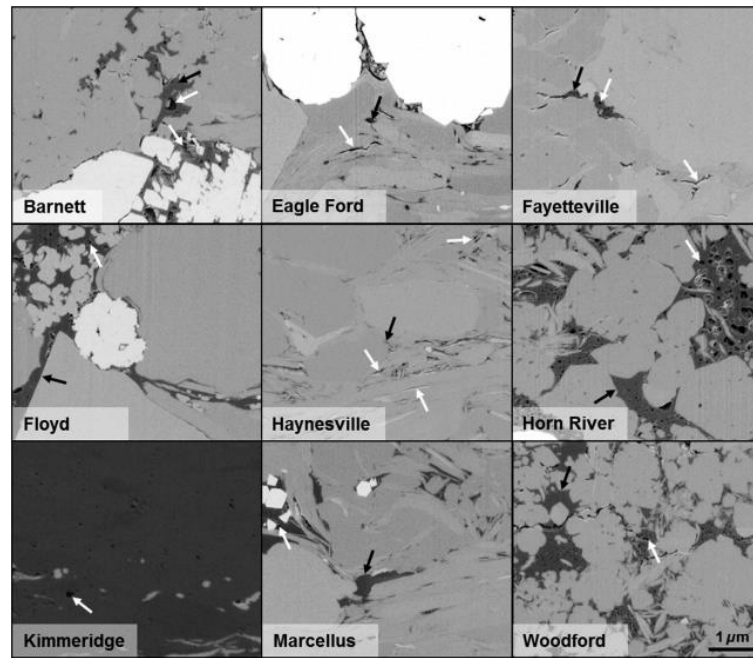


Figure 2.1 — Backscattered SEM images of various 4.8 μm x 4.1 μm shales.(Curtis et al. 2012)

The hydrocarbons trapped inside of shales have come to play an increasingly important role in meeting natural gas demand. Because the majority of these resources occur in formations of very low permeability that are generally inaccessible (in terms of economic production) without fracturing, understanding the fracture behavior of shale has likewise become an increasingly important area of study. One type of data that may be used to improve the predictions of subsurface changes after a fracture treatment is shale's microstructure and nanostructure. This complexity raises the possibility that assumptions in any models that regard shale strictly as a macroscopically homogeneous material may result in various inaccuracies in both lab experiments and in field studies. Thus, without a deeper, physics-based understanding of the possible effects of microscale structures on

the initiation and propagation of fractures in shale, the reliability of fracture prediction models may suffer if such information is not considered.

2.2 Shale Structure and Fracturing Relationships

There have been many studies relating the macroscopic composition of shale to its mechanical properties. A consistent finding amongst multiple groups and experiments have noted that the fracture toughness and elastic modulus of shale decreased significantly when their kerogen content (Closmann and Bradley 1979, Grady and Hollenbach 1979, Chong et al. 1984, Shitrit et al. 2016) increased. Others have taken a closer look at the quartz content of shales. While one may naively assume that the addition of a stiffer component would result in a stronger bulk material, many studies seem to suggest either a weak negative correlation or no correlation between quartz content and shale strength (Gunsallus and Kulhawy 1984, Koncagül and Santi 1999, Barbour and Ko 1979). This led many to believe that it is the structural arrangement of quartz that can result in any measured macroscopic response. For example, Koncagül and Santi (1999) went as far as to suggest that the quartz content itself was a poor predictor a rock's uniaxial compressive strength (UCS), and that the interlocking of quartz particles was what determined the UCS.

Such macroscopic behavior can be explored in greater depth by examining the microstructure of these materials and how changes in the microstructure can create a macroscopic change in the mechanical properties of materials. In fact, the relationship between a material's microstructure and its macroscopic properties is studied in depth in a variety of fields, including chemical, aerospace, mechanical, petroleum engineering and

geology. For example, in a parametric simulation study of microscale heterogeneity on the macroscopic deformation of rocks, Blair and Cook (1998) created local perturbations in otherwise perfectly arranged triangular lattices of materials to show how such perturbations lowered the mean ultimate tensile strength of a material. More recently, Kumar, Sondergeld, and Rai (2012) studied the relevance of shale microstructure to its elastic moduli by performing nanoindentation experiments in shale samples from various formations, and concluded that there was a positive correlation between quartz/carbonate content and the elastic modulus of shales. They even provided empirical formulas for shales ranging in elastic moduli between 20 GPa to 80 GPa and concluded that the mineralogy, clay content, total organic content, and porosity all played a role in the Young's moduli of materials. Similar relationships between composition and mechanical properties have been proposed from other studies (Sone and Zoback 2013). However, Ambrose (2014) noted that such correlations show a great deal of scatter and did not observe any clear, convincing trends in these relationships.

2.3 Microscale Properties

To accurately capture the fracture at the microscale, one must recognize that the properties of macroscopic materials incorporate pre-existing flaws that should be explicitly captured from microscale images. Thus, unlike macroscopic simulations, any microscopic fracture simulation should consider the elevated moduli and fracture strength of materials in a microscale simulation. Some of the most significant changes in the mechanical properties that will be caused by the characteristics of the microstructure, i.e., the moduli of the

constituent materials of the shale, fracture stress of the constituent materials, and the particle distribution (size and concentration of the occlusions of the different materials) within the shale structure.

In microscale and nanoscale experiments, one of the most well-known effects is that pure materials appear to have a greater elastic moduli and fracture stress under nanoindentation when compared to the macroscopic counterparts (Mukhopadhyay and Basu 2007). The specific reasons for this effect vary depending on the specific materials and are under investigation, but it appears that the “smaller is stronger” size effect applies to the constituent materials of shale, as is the case in other materials. This has fueled an interest in estimating both the moduli of elasticity and the yield strength of the constituent materials of shales through the use nanoindentation experiments (Kumar, Sondergeld, and Rai 2012). It should be noted that nanoindentation data often show a large range of responses regardless of the specific materials or methods used due to the limitations of nanoscale mechanical experimentation and the inherent heterogeneity of the structure. Additionally, different publications can report very different mechanical properties because of differences in their individual experimental methods and procedures, as well as in the specific shale samples they used. For example, one set of nanoindentation experiments on Woodford Shale samples resulted in an estimate for the clay component’s elastic modulus as 20.323 GPa with a standard deviation 2.927 GPa (Bennett et al. 2015). In a different set of measurements of shale material layered on a glass substrate, it was found that pure kaolinite had a dry Young’s modulus of elasticity of 2.59 GPa, within a deviation of ± 0.16 GPa (Bathija 2009). From their nanoindentation experiments on

Marcellus shale, (Mason et al. 2014) concluded that their results and observations are vulnerable to various experimental artifacts caused by the difficulty in isolating mechanical behavior to the head of an indenter tip, thus yielding far higher or lower stiffness measurements than the true stiffness of the material.

Despite these drawbacks, literature nanoindentation data can provide valuable insights into the realistic range of moduli one should expect for the constituent material of shales. While different testing methods, sampling techniques, and procedures suggest different absolute values of the moduli of elasticity, they often agree in their assessment of the relative mechanical strength of each constituent component in an aggregate material. What is clear from various studies is the quartz, feldspars, and pyrite materials in shales have significantly higher elastic moduli than clay.

Less easily explored are the fracture stresses of the constituent materials of shales. It is believed that much of the fracturing in brittle materials tends to occur as a result of tensile or shear stress because even macroscopic compressive stresses can exert local tensile stresses due to heterogeneities in a system (Blair and Cook 1998). Thus, it may be difficult to properly calibrate microscale fracturing simply because there have not been many experiments that have actually focused on estimating a material's tensile or shear strength. In many cases, the only available information on the possible range of failure criteria for shear and tensile failure in microscale materials can only be provided by available macroscopic material property information enhanced by available microscale indentation information. For example, the bulk material properties can be used as the lower bound of failure strength, and the estimated compressive failure strength from hardness tests as the

upper bound. Alternatively, one may attempt to use estimates of the material's failure envelope with the estimated compressive strength from indentation tests. However, both methods suffer from the approximation of applying properties that may be relevant only in a macroscopic setting to a microscale setting.

Despite the difficulties in studying fracturing at this size-scale, however, recent experiments performed on shale using Environmental Scanning Electron Microscopy (ESEM) provide further evidence that the microstructure of shales can affect fracture behavior by recording this fracture as it occurs in humidified environments. In one experiment involving two samples at 2.2% hydration and 10.2% hydration, Wang et al. (2016) demonstrated that the more hydrated shale has a lower elastic modulus, a lower peak stress, and the nucleation of microcracks is far more pronounced than those in the drier sample (**Fig. 2.2**). As shown in **Fig. 2.2**, the contrast from the greyscale backscatter SEM image can be used to track different minerals in the material as it experiences deformation. The result of this experiment shows clear evidence that the quartz and carbonate occlusions in the shale samples modify the direction and trajectory of the fracture, as shown by the color from blue to red.

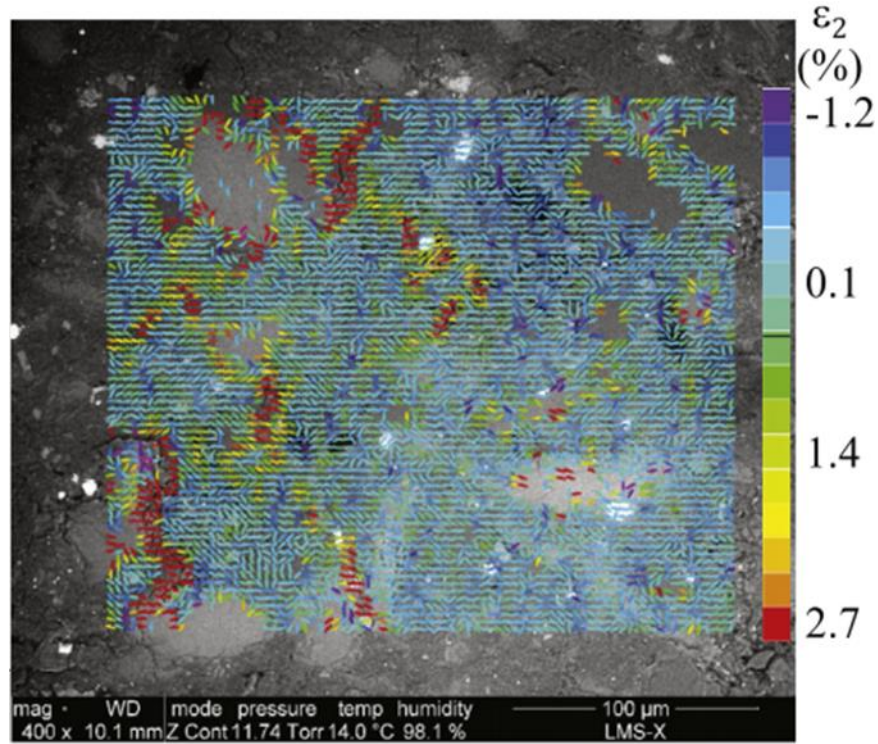


Figure 2.2 — ESEM experiment showing the microcracks of a deformed and hydrated shale sample. Most microcracks are reported to occur in the fracture-matrix interface (Wang et al. 2016)

2.4 Discrete Element Model and Implementation

Using the discrete element method (DEM) to investigate the fracturing of brittle aggregate materials such as concrete and rocks has been used in various studies due its ability to simulate the behavior of granular material and effectively handle discrete events such as fracturing (Scholtès and Donzé 2012, 2013, Duriez, Scholtès, and Donzé 2016). In a 2D study of concrete, Kozicki and Teichman (2007) studied a two-material system, comprised of a weaker cement and stiffer aggregate material, with an intermediary transition material to model the Interfacial Transition Zone (ITZ) between materials. Most importantly, they demonstrated that micro-cracking in materials can create non-linearity in the bulk

response to material deformation, a response that may occur before and after achieving any ultimate system stress due to the aforementioned micro-cracking in the simulated volume. I will thus adapt the methods described in previous DEM studies by associating particles in a DEM simulation with the materials identified from the available EDS/SEM data.

DEM was originally developed for soil applications and granular assemblies (Cundall and Strack 1979). This method involves rigid elements that are assumed to deform very little in shape relative to displacements in the system as a whole, and describes the deformation processes as interactions between the elements (**Fig. 2.3**). In the DEM model, all materials are comprised of elements (spheres or polyhedral) and the interactions between them. In mechanical simulations, the normal and shear moduli are frequently estimated by assuming interactions are spring-like, as labeled in the figure above by a tensile modulus k_N and shear modulus k_s (Stránský, Jirásek, and Šmilauer 2010). The direction and force of this interaction on each element is determined by various constitutive laws. In comparison to continuum approximation models of standard geomechanics (such as the finite element method), such a model is innately suited for discontinuous problems such as fracturing (Stránský, Jirásek, and Šmilauer 2010).

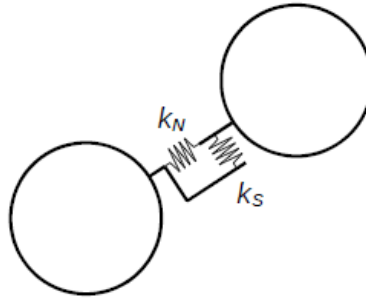


Figure 2.3 — DEM models all materials as an assembly of elements. Two commonly seen element interactions are the normal and shear elastic interactions, controlled by a tensile modulus k_N and shear modulus k_S (Stránský, Jirásek, and Šmilauer 2010).

It is important to note that, despite its advantages in simulating discontinuous events such as fracturing, the DEM also suffers from drawbacks that must be considered during calibration. Of these considerations, the most important is to consider the effect of the individual element size. As indicated by (Scholtès and Donzé 2013)., the number of elements can bias the estimates of the effective elastic modulus and of the Poisson's ratio. They also recognized that traditional closed-packing spherical structures do not possess an altogether optimal behavior to simulate nonlinear fracturing, and noted this need can be better accommodated by increasing the interaction radius of spheres beyond their contact points.

As is obvious, the desired properties and macroscopic behaviors in the DEM approach are determined by the physics and length scales of the element interactions. One may be tempted to account for the plasticity of a material by explicitly including plasticity in the physical interaction between particles, but doing so would neglect that the simulations are supposed to explicitly capture the small fractures which cause this macroscopic plasticity. Thus, in order to minimize the computational resources required to ensure that fractures

are captured, the goal of any DEM simulation should be to minimize the number of phenomena to be simulated directly between inter-particle bonds (i.e., to minimize the number of calculations between elements) while still being able to macroscopically capture the desired behavior. Because I only am attempting a qualitative approximation of fracture behavior, I will avoid explicitly including plasticity in the particle bonds because this would likely be of far greater benefit when quantitative accuracy is a priority.

The specific implementation of DEM I use will be the open-source framework Yade, which has many useful features and one significant drawback for scientific computing purposes (Kozicki and Donzé 2009). The underlying computational algorithms of Yade are written in C++, though end users would use Python to set up the simulation and perform data collection. This package contains the option to implement any additional physics and interactions, though a pre-built elastic particle model was specifically chosen for this project due to its data collection features. However, while Yade is easily configurable and extensible, the unmodified version is only viable for simulations involving relatively small numbers of particle (e.g. $<10^6$) because the simulation cannot be run on parallel processors. Fortunately, for a 2D simulation with a qualitative focus, a relatively low-particle count is an acceptable limitation. To see the code or instructions on how to run my simulation, you may look at the code and instructions provided in Appendix B.

CHAPTER III

DEVELOPMENT AND IMPLEMENTATION OF METHODS

3.1 Motivation for Data Conversion and Simulation

The overarching purpose of the methods used or developed in this project is to make use of microscale shale structure image data to better understand shale fracturing in microscale mechanical simulations. The EDS images allow the determination of the position of atomic elements in microscale shale samples. Combined with an understanding of the mineralogy of these shale samples, it is possible to map the specific position and shapes of the various components/minerals in the shale. This map allows the representative (and assumingly correct) assignment of the generated elements in a DEM simulation to the appropriate materials, thus imparting them their properties and interaction specifics. By manipulating the element-scale properties and interactions, the model is then calibrated to capture the relevant macroscopic mechanical properties of fracture stress and elastic modulus in multi-component materials.

After calibrating the properties of each material individually, I can study aggregate fracture behavior by applying boundary stress and strain conditions in a simulation with different materials. A significant advantage of this simulation approach is the ability to modify the model inputs and modes of element interaction, thus helping illuminate poorly understood phenomena that are difficult to capture experimentally, e.g., the effect of material interfaces and quartz occlusions, on the fracturing behavior of shales.

3.2 General Data Conversion and Simulation Process

In order to use the available data from the four Niobrara samples, I must convert all SEM and EDS images into a microstructure, and then simulate deformation on this microstructure. The general steps to this process are:

1. Ensure all SEM and EDS images are the same size by either a pre-processing step or re-sizing the image.
2. Convert all EDS and SEM images into binary arrays to define the presence or absence of atoms or void space in every pixel of the images.
3. Use a pre-defined set of rules that determines the identity of the material at each pixel from these binary arrays.
4. Randomly generate a set of packed 2D particles in a rectangle and use the material map to assign each particle with its material based on the particle's position.
5. Ensure that all cohesive interactions are in place and apply desired boundary conditions.
6. Run the DEM simulation.

3.3 Material Properties

The scale of the simulations in this study (as described by the element sizes) is such that reliable data on the properties of pure materials are difficult to acquire. While nanoscale and microscale indentation experiments are able to provide certain values, the mechanical properties of pure clay are especially difficult to capture.

The values (obtained from the literature and assumed ones) I used in the DEM simulations and the corresponding sources are shown in Table 3.1. As noted in the literature review (see Section 2.3), the mechanical properties of mesoscale materials is heavily dependent on the methodology and size/scale of the material being measured. In these simulations, of primary interest are the material properties of three particular minerals known to occur in Niobrara shales: kaolinitic clay, carbonate, and polycrystalline quartz. I realize that kerogen may play an important role in various facets of hydraulic fracturing, but the significant weakening of oil shales at elevated temperatures, when combined with studies that show room-temperature shale is as weak if not weaker than clay, suggests that kerogen may have a very low elastic modulus at subsurface conditions (Lempp et al. 1994, Bennett et al. 2015, Kumar et al. 2012). Thus, I will approximate kerogen as void space in my mechanical deformation tests (i.e., the subject of simulations).

Table 3.1 — Mineral Material Properties Used to Calibrate DEM Bond Parameters

Material	Property	Value	Source
Clay	Elastic Modulus	4-20 GPa (calibrated to 5 GPa)	(Wang et al. 2016)
	Poisson Ratio	0.144	(Bathija 2009)
	Tensile Stress At Fracture	96 kPa	(Chenu and Guérif 1991)
	Friction Angle	10	(Wang and Li 2014)
Carbonate	Elastic Modulus	80 GPa	(Wang et al. 2016)
	Poisson Ratio	0.25	(Yale and Jamieson 1994)
	Tensile Stress At Fracture	1 MPa	(Vásárhelyi 2005)
	Friction Angle	34.3	(Wang and Li 2014)
Quartz	Elastic Modulus	96 GPa	(Wang et al. 2016)
	Poisson Ratio	0.11	(Greaves et al. 2011)
	Tensile Stress At Fracture	61 MPa (unpolished natural quartz)	(Chao and Parker 1983)
	Friction Angle	23	(Wang and Li 2014)

3.4 Mechanical Model

While I would ideally want to consider all available material properties available and capture the entire spectrum of the desired physical parameters, I must also consider the corresponding computational demands and the lack of data that exists in literature. I thus ignore considering effects of bond plasticity, non-linear elasticity, and compression fracture between bonds. For fracturing behavior, the most important parameters are ultimately related to the various limits that will result in bond failure between particles.

Because of this observation, I thus concentrate the calibration efforts on properly capturing the Young's modulus and the tensile/shear fracture stress.

In the simulations, I approximated all three materials as brittle, in which interactions are elastically loaded until a critical stress breaks the bond. While this linear elastic fracture mechanics (LEFM) model may not be appropriate for kerogen, the LEFM model is frequently applied for ceramics and crystalline materials such as clay, quartz, and carbonate (Brooks, Ulm, and Einstein 2013, Jaya, Kirchlechner, and Dehm 2015, Akono and Kabir 2016). In general, I calibrate the inter-particle bond parameter values so that uniaxial tensile experiments can match the material properties determined from Table 3.1, first by modifying the bond's Young's modulus parameter and then by modifying the bond fracture stress.

3.5 EDS Pre-Processing and Mapping Methodology

Without appropriate information on the shale microstructure, no simulations or model system would be possible, let alone useful for any systematic study of the importance of the shale microstructure. Such data are difficult to approximate from traditional macroscopic studies, but elemental analysis provided by EDS scans generously provided via personal communication from Lawrence Berkeley National Lab can be used to quickly and accurately determine the microscale composition of shale (Votolini and Ajo-Franklin 2016 -- *Personal Communication*). Using **Fig. 3.1** as an example, one can begin with an original EDS scan such as the 82 μm x 56 μm image shown on the left. By recognizing that black represents positions that have no calcium, red is the position with calcium, a

filter can be used to generate a binary map which extrapolates whether calcium is present in an equivalently-sized region. This can then be used to generate the binary map shown on the right, where blue pixels represent the absence of calcium and red regions denote the presence of calcium.

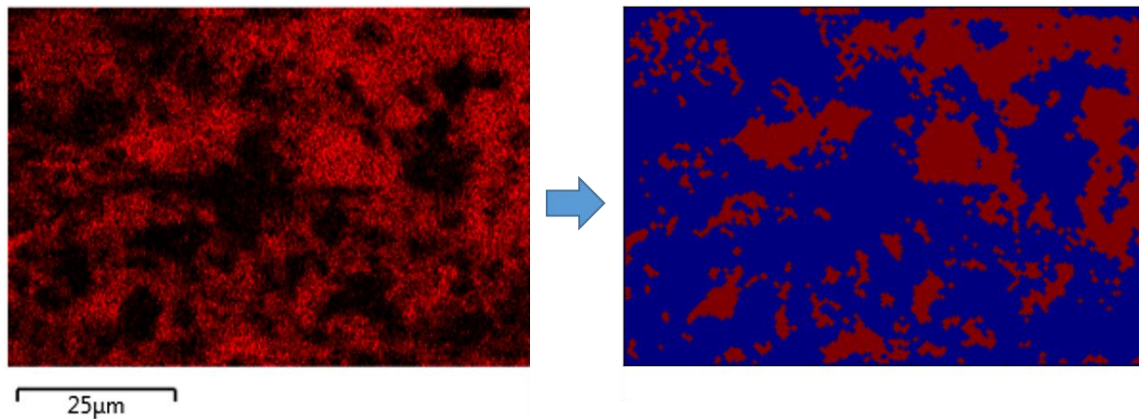


Figure 3.1 — EDS digital output from converting a $82\text{ }\mu\text{m} \times 56\text{ }\mu\text{m}$ EDS image showing calcium on the left (black = No Calcium, red = Calcium) into the binary image on the right (Voltolini and Ajo-Franklin 2016 -- *Personal Communication*).

In the case of Niobrara shale, the working assumption is that the possible materials are: kerogen or void space (treated as the same in the geomechanical simulations because of the very low mechanical strength of high temperature kerogen), clay (kaolinite), carbonate, quartz, or pyrite. By using Python's Scipy image processing package on the EDS images (Jones, Oliphant, and Peterson 2001), the regions where a particular element from the periodic table is present can be readily identified. By combining the output of these scans with user-generated rules on how the materials are identified and determined, a map can be created that will assign a specific material to every pixel of the EDS scans.

The decision process that provides the rules for element and material assignment to the pixels of the EDS images is shown in **Fig. 3.2**.

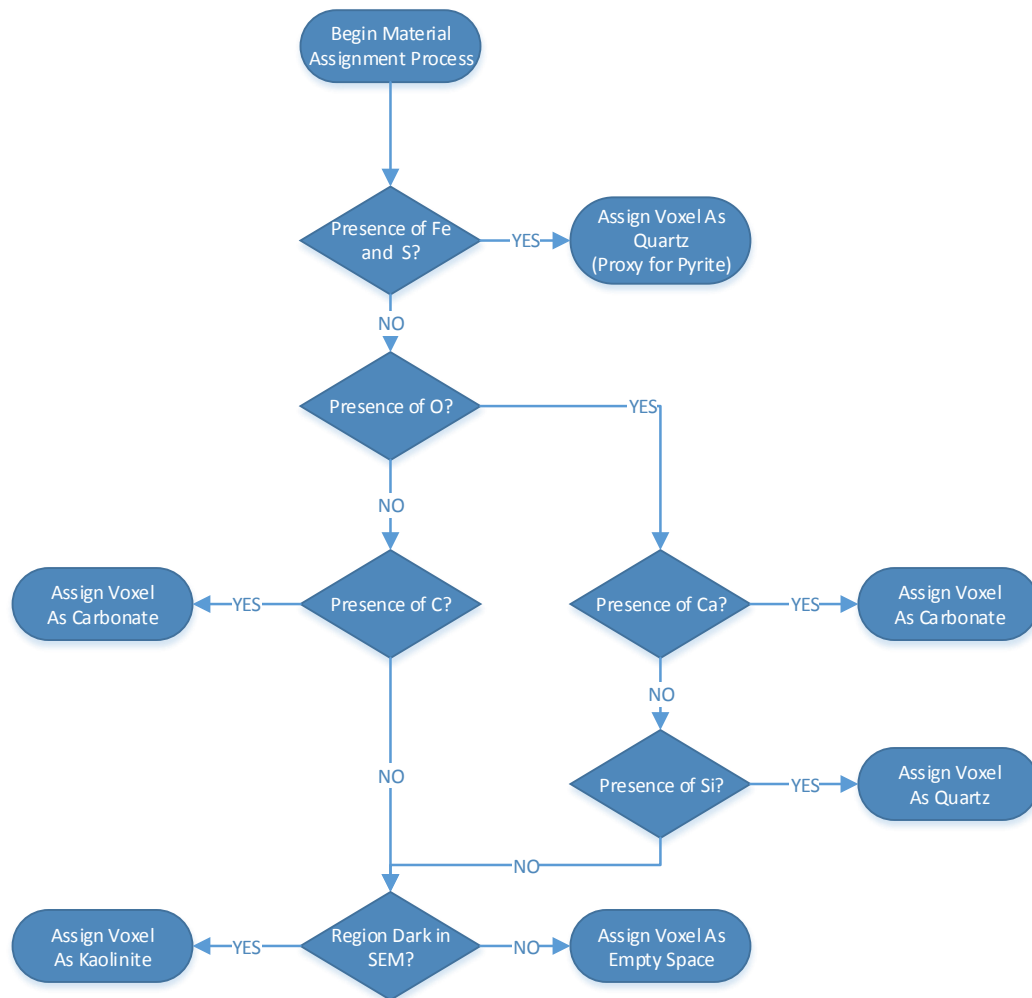


Figure 3.2 — Rules that were implemented in order to define the material properties based on EDS information.

By applying these rules for every pixel we can generate a complete map to determine the identity of each material in every map. For example, using the various EDS images and SEM information available to us for the first set of images, we end up with the material definition map shown in Fig. 3.3

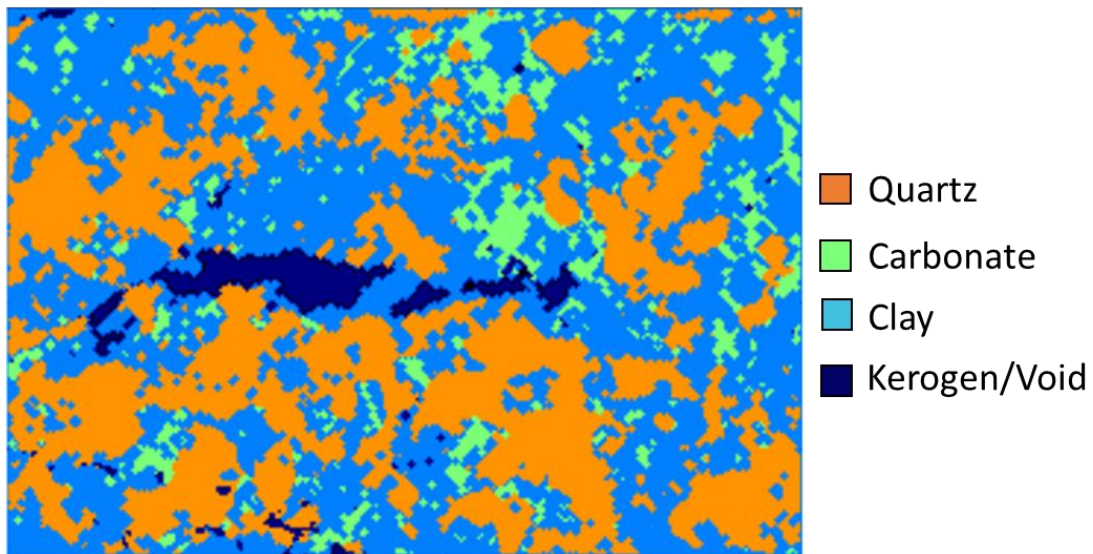


Figure 3.3 — Full 2D material for Sample A. This map is generated from the rules outlined in Fig. 3.2 to help with assigning material properties for simulation purposes.

I further modified this material map to account for the interfaces between different materials through the use of the open-source computer vision library OpenCV as shown in **Fig. 3.4** (Bradski and Kaehler 2008). By being able explicitly define an outline to every shape, we can monitor the relationship between the interface of each shape and any fracturing that may occur. This will make later analysis more convenient, as well as provide more options to directly manipulate the boundary interfaces between types of materials.

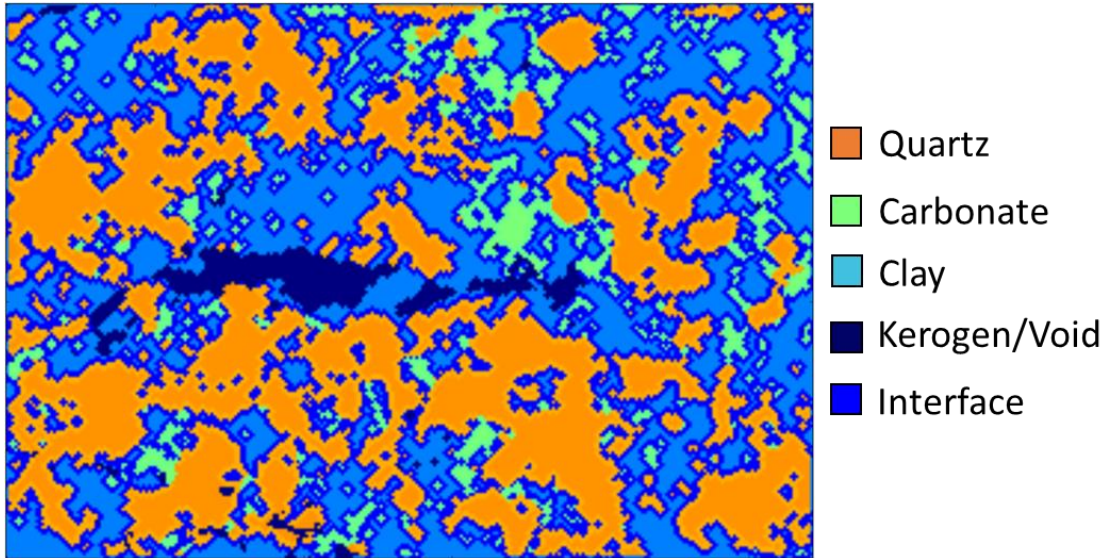


Figure 3.4 — The final material map keeps track of both materials as well as whether or not a pixel is on the interface between two materials.

The same process can be applied for each of the four Niobrara shale samples used in this study. As shown in **Fig. 3.5**, there is a great deal of variety in structure and composition of the shale microstructure. While Samples A and B are primarily composed of clay, samples C and D included areas where the composition was dominated by carbonate or quartz. In general, this process should be applicable for any combination of EDS/SEM images, though it is strongly advised that one consult an expert to ensure that no unexpected minerals are encountered and mislabeling does not occur.

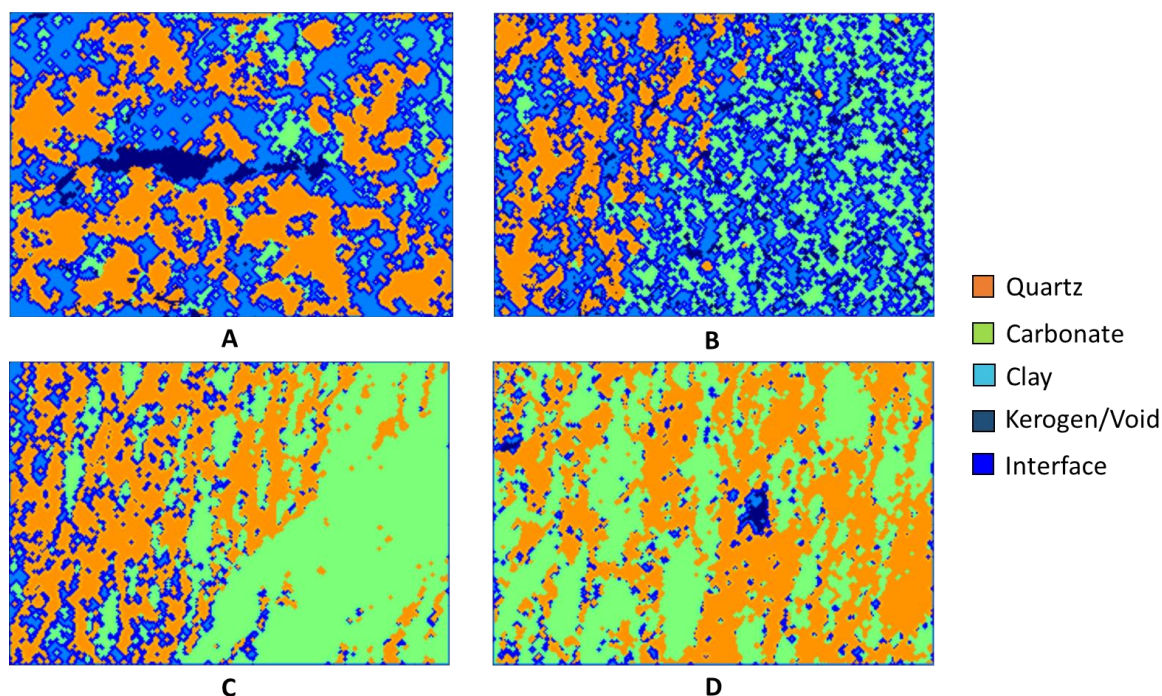


Figure 3.5 — Material maps for all four samples. The dimensions of each sample are, respectively, 81 μm x 56 μm , 220 μm x 150 μm , 1400 μm x 990 μm , and 520 μm x 360 μm

3.6 2D Particle Generation Methods

In the methodology I am proposing in this study, the creation of the per-pixel 2D map from the original SEM/EDS data is followed by the generation of the corresponding DEM network of elements. Generating the tightly-packed but randomly placed elements in a 2D DEM simulation can be done from either a gravity-settling or particle growth method. In the first method, a given element size is pre-selected using the pixel length as the normalization coefficient. I then simulate the gravity deposition of these particles by allowing the particles to fall into the desired simulation area. All spheres outside of this desired area are excised and the remaining spheres are exported to form the simulation particles. In the growth method, a set number of elements are randomly placed in the simulated space and then allowed to uniformly grow and move, eventually pushing against the boundary walls, until the stress at the boundary walls reaches a user-defined threshold. The primary difference between the two methods is the way small voids from imperfect packing manifest themselves, resulting in slight differences in porosity. In the gravity-settling method, the void spaces are concentrated at the top of the DEM element network, while the element growth method distributes these flaws randomly throughout the entire network of (**Fig. 3.6**). While the growth method does result in a higher porosity, the macroscopic difference between the two resulting porosities decreases significantly when the total number of particles increases.

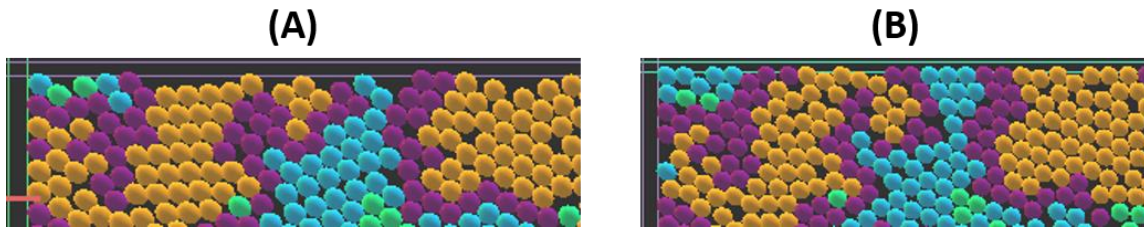


Figure 3.6 — Example of differences between gravity-settling particle growth packing. (A) the gravity growth model concentrates irregularity at the top but is more tightly packed (B) The growth model distributes irregularity throughout the model, but is more uniform at all boundaries.

To better understand potential differences between the gravity-settling method and a particle growth method, I estimated the modulus and fracture stress using the calibration method described in section 3.7. As shown in **Fig. 3.6**, the gravity settling method results in tighter packing in the interior of the mesh, which seems to slightly increase the Young's modulus and the fracture stress when the number of particles is kept constant. However, the range of values and the corresponding standard deviation of the estimates in the gravity settling method are also larger, which may be due to the concentration of heterogeneities at the edge of the geometry. An additional shortcoming of the gravity settling method is that the time it requires for the element/particle placement is approximately 4-5 times greater than that of the particle growth method. Thus, all simulations tests following this calibration process used DEM assemblies generated exclusively by the particle growth method.

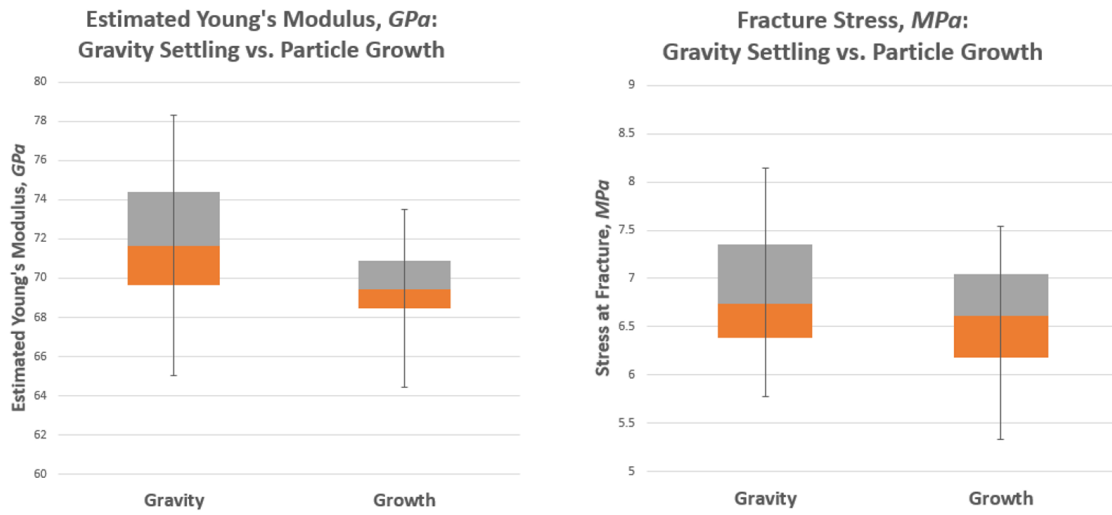


Figure 3.7 — Box and whisker plots showing differences in fracture stress and Young's Modulus estimates. 40 trials were run for each parameter in a simple $20\ \mu\text{m} \times 30\ \mu\text{m}$ uniaxial tension experiment for a homogeneous material. Grey and orange shows the first quartile above and below the median, respectively, and whiskers show the absolute maximum and minimum Young's Modulus found. The growth method is slightly more accurate and takes far less time to generate an assembly of particles.

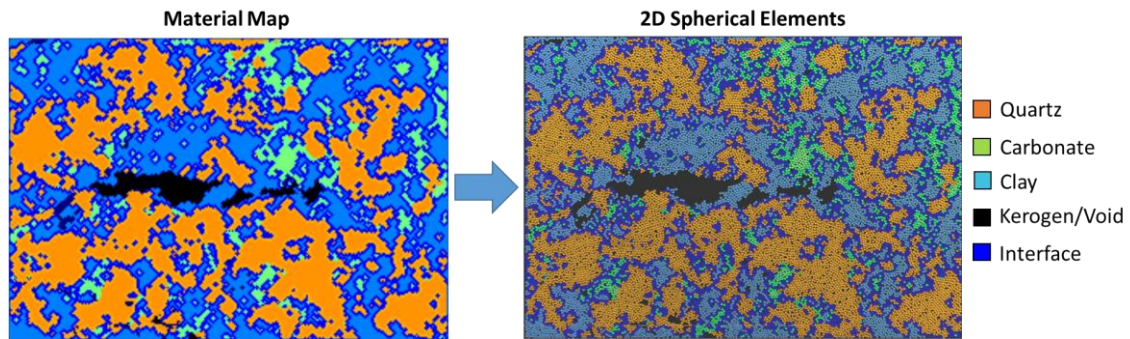


Figure 3.8 — Image showing conversion of map to particle assembly. By using a material map such as the one for Sample A on the left, I can assign a material identity to every particle in the DEM simulation. In the image on the right, a 56,000 particle simulation is created by determining a particle’s material properties from its position.

Once both the material map and DEM elements have been acquired and defined, each sphere in the model is associated with the material occurring at the location of the center of the particle (**Fig. 3.6**). While it is possible to arbitrarily choose the number of particles used in my simulation, it is evident that using particles with large radii will result in under-representing smaller features or distorting their shape in the final model. The selection of the number of particles is guided by the desire to have at least one interaction reflecting every distinct occlusion identified, though for computation time reasons this may not always be feasible.

3.7 YADE-DEM Material Property Calibration

DEM simulations require appropriate calibration in order to properly simulate the behavior of materials under mechanical stress (Wang and Alonso-Marroquin 2009). The microscale parameters that govern the interactions between the particles in a DEM simulation and the macroscopic material properties of the system as a whole are heavily

correlated but not equal to each other. For example, if one wished to simulate a rectangular block of material with a measured Young's modulus of 100 GPa, they would need to elevate the Young's modulus of the individual bonds because the bond surface area does not completely cover the cross-section of the simulated volume and most of the bonds will not be parallel to the direction of stress.

Thus, before further work can be done in an aggregate simulation, the relevant macroscopic material properties must be captured. Because a primary goal of this study is to properly capture fracture shape and behavior, significant effort is focused on correctly capturing the Young's modulus and the tensile fracture stress of each individual component. Because this study focuses on recreating microscale structures with nanoscale data, it would be ideal to have access to confined microscale experiments (if such tests exist) designed to provide estimates of the normal moduli, shear moduli, critical tensile stresses, and critical shear stresses of each material of interest. However, such data could not be found in the literature. Consequently, I directly calibrate bond properties in order to achieve the two material properties that were most easily collected from the literature: the Young's modulus and the critical fracture stress in table 3.1. The Poisson's ratio of the bond was left as its original value in Table 3.1, while the critical shear stress was always set to be half that of the bond modulus. Note that the approximations in the parameter values, necessary because of the lack of data, may affect the quantitative predictive ability of this model. Even if this the case, this does not diminish or invalidate the conclusions and does not affect the importance of the development of the methodology I propose, which is a key contribution of this study.

I chose an iterative method to calibrate the bond properties in this simulation. I created a simulated area of $20\ \mu\text{m} \times 30\ \mu\text{m}$ under the assumption that each pixel would be a $1\ \mu\text{m} \times 1\ \mu\text{m}$ pixel. As will be discussed in section 3.8, I populate the simulation with one element for every hypothetical pixel in the simulation using a particle growth method. I then run a simulation in which I apply uniaxial tensile loading to the cohesively-joined 600 particles and analyze the output from these simulations to determine the elastic modulus and fracture stress. By repeating this test in a batch, I can calculate an average value for both the Young's modulus and fracture stress. Finally, I can repeat the entire process after modifying the bond Young's modulus, tensile fracture stress, and shear fracture stress, which is always set to be half the tensile fracture stress.

3.8 Effect of the Particles Count on Simulated Material Properties

The parameters that can influence the outcome of my simulations can be divided into three different categories: element/interaction properties, boundary-condition parameters, and element geometry parameters. The first two sets of parameters have physical analogues, either because they are directly related to the macroscopic material properties or because they reflect the boundary conditions that are to be simulated, while the third set of parameters must be calibrated to balance and optimize the simulation performance against the simulation accuracy. Such parameters include the element radius (or conversely, the total number of elements in the simulated space), the method of generating the mesh (gravity-settling vs. particle growth), the force dampening effect, and the initial element interaction length. Just as in any other simulation method, it is imperative to establish that

this third set of parameters is appropriately understood so that they do can be appropriately accounted for when interpreting any of the simulation results.

Of the element geometry parameters listed, the only one that has any significant effect on the measured mechanical properties is the particle count parameter. To study this effect, I altered the number of particles that were made to simulate tension in a $20\ \mu\text{m}$ x $30\ \mu\text{m}$ area assuming that each $1\ \mu\text{m}$ x $1\ \mu\text{m}$ square was captured by a pixel. . As shown in **Fig. 3.8**, the results of my single-material calibration tests for the estimation of Young's modulus of elasticity demonstrated the positive relationship between simulation precision and the number particles of a simulation. This is in agreement that this is in agreement with the known observation that the precision of a material property estimate in a DEM simulation would approach a limit as the number of particles within the cell approached infinity (Stránský, Jirásek, and Šmilauer 2010). In addition, I performed a similar to study to explore the effects of particle count on the tensile fracture limit, as shown in **Fig. 3.9**, and reached the same conclusion.

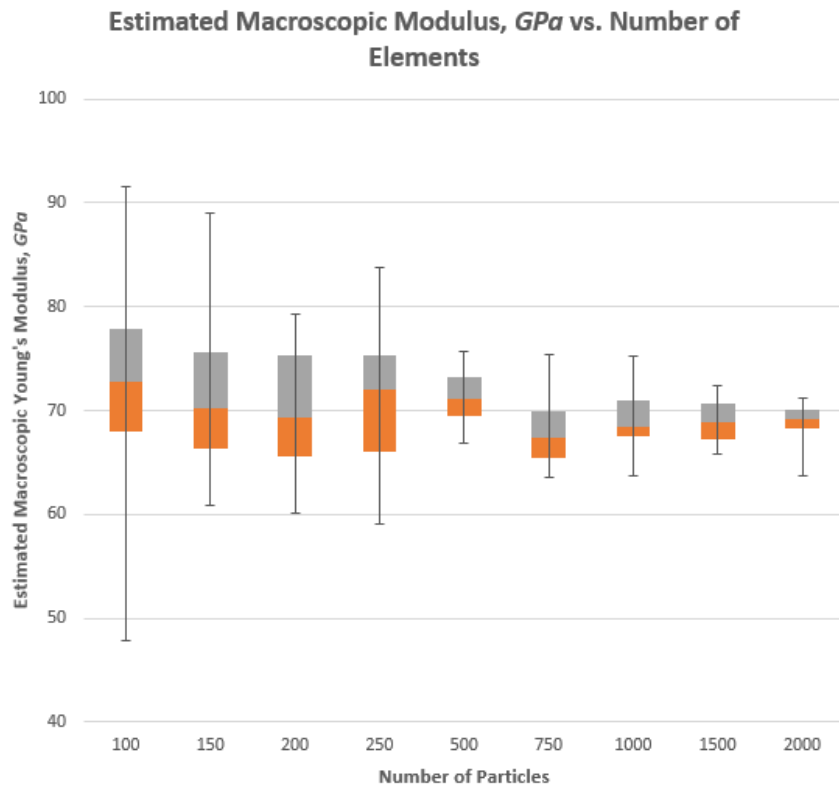


Figure 3.9 — Box and whisker plot showing Macroscopic Young's Modulus vs. the number of particles. 20 trials were run for each parameter in a simple uniaxial tension experiment. Grey and orange shows the first quartile above and below the median, respectively, and whiskers show the absolute maximum and minimum Young's Modulus found. Increasing the number of particles only slightly increases calibration accuracy once the number of spheres approaches the original number of pixel.

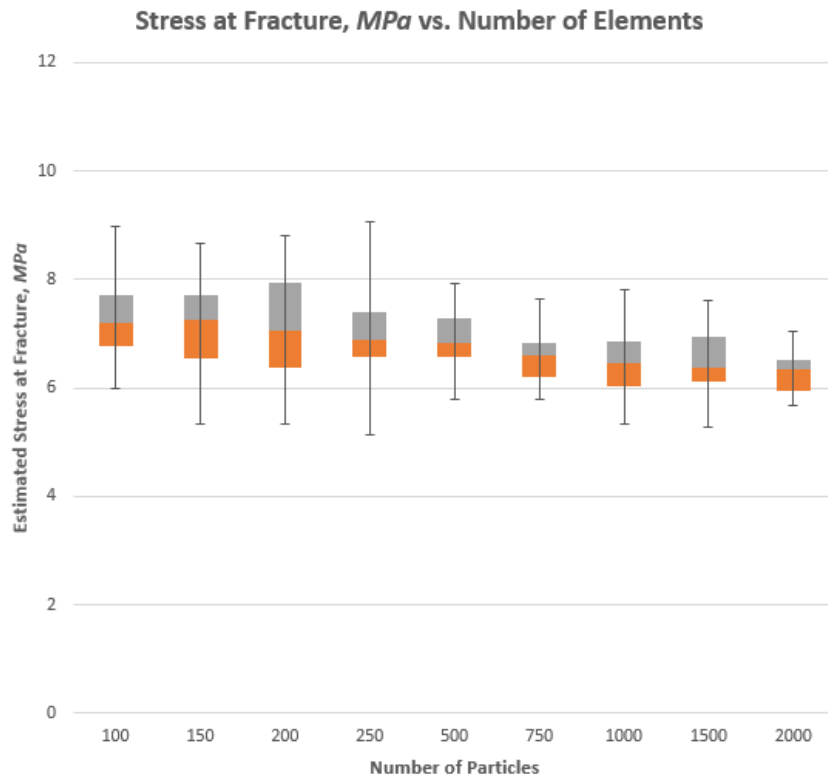


Figure 3.10 — Box and whisker plot showing fracture stress vs. the number of particles. 20 trials were run for each parameter in a simple uniaxial tension experiment. Grey and orange shows the first quartile above and below the median, respectively, and whiskers show the absolute maximum and minimum Young's Modulus found. Increasing the number of particles only slightly increases calibration accuracy once the number of spheres approaches the original number of pixel.

From the evidence provided by the calibration process discussed above, it is incumbent upon me to ensure that the number of particles in the simulation approaches the original pixel count of each starting image the simulation. Because it is not possible to dismiss the possibility that the particle count exhibit some bias toward the macroscopic material properties, it is important to recognize their impact on the simulation results and to compensate for their effect accordingly. In practice, this means that I must ensure that the number of particles for a simulation is equal to the number of pixels used and that I do not deviate from this ratio when I compare the mechanical properties of different samples.

3.9 Quantifying and Visualizing Fracture Events

One of the key objectives of this study is to determine the relative effect of mineral occlusions on the fracture pathway/trajectory. To accomplish this, it is imperative to have a method to quantify these very irregularly-shaped occlusions. While there are different methods to classify and quantify the shape of materials, the simplest approach in this case is to consider effect of occlusion shapes on the number of particles defined as an interface particle. As the number of occlusions increase, I would expect the number of interface particles to increase. I can compare this value to the ratio between the number bonds broken at one of the interfaces and the total number of broken bonds.

Fortunately, both the counting of interface particles and the categorical counting of fractured bonds is possible in Yade. The fracture locations are determined and mapped in the original structure (**Fig. 3.10**). This visualization is performed by collecting the list of fracture event coordinates, which can be conveniently accessed from the jointed cohesive

fracture model in Yade. As will be expanded upon Chapter 4, the location of these fractures suggest a strong relationship with the interface of different occlusions. Most importantly, because of the ability to define interface particles in either the clay or mineral occlusions, it is possible to determine the specific type of materials being fractured and whether the fractures are occurring near the occlusion interfaces within clay, within the body of clay that is not connected to any occlusions, or in the interior of occlusions.

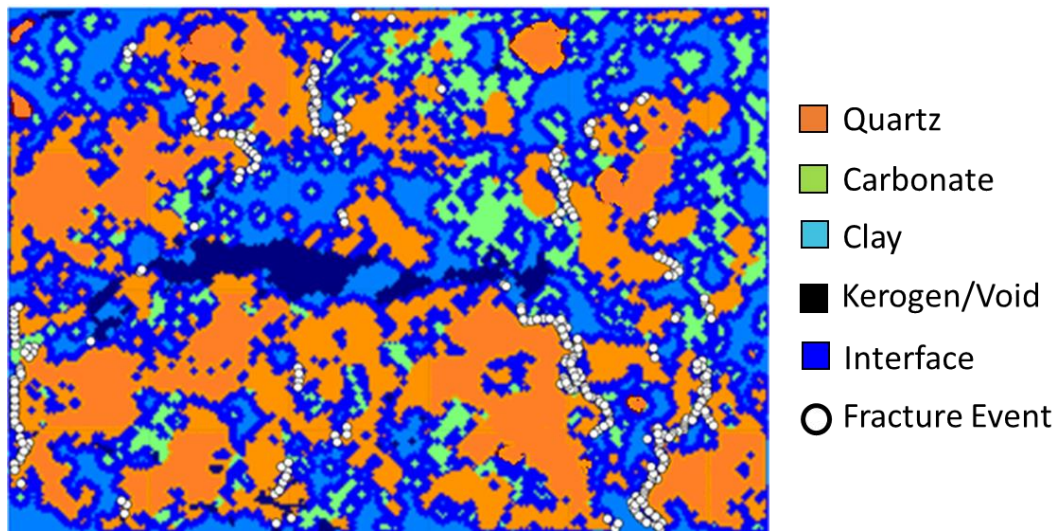


Figure 3.11 — The material map for sample A. Each color represents a different mineral or compound.

CHAPTER IV

UNIAXIAL FRACTURE STUDY

4.1 Uniaxial Test Overview

Due to its simplicity, the uniaxial fracture test is one of the most common tests used to obtain information about fracture properties. While 2D uniaxial fracture tests are not likely to provide the quantitative properties necessary to optimizing hydraulic stimulation in ultra-tight shales, these tests are capable of providing qualitative insight into the way aggregate materials fracture. Moreover, the data provided from the four samples is best suited for 2D simulations and uniaxial tests provides a simplified means of studying the resulting microfractures as a function of applied stress and internal microstructure.

Thus, using the tools and techniques described in Section 3, I simulated uniaxial mechanical deformation experiments on shales in order to study the effects of different microstructures on the resulting 2D deformation. Specifically, I was interested in the effects that mineral occlusions in the shale may have on the effective Young's modulus of elasticity, the fracture stress, and the bias towards interface boundaries. My ultimate goal was to demonstrate the importance or unimportance of the microstructure (and especially of the occlusions), as defined by the location and the shape of the failing interfaces (i.e., characterized by breaking bonds) between different constituent materials. To accomplish this, I conducted simulations of uniaxial tests on 2D shale microstructures with and without occlusions, with particular emphasis on confined compression of the DEM particle assembly in order to develop a better intuitive understanding of the effects of shale microstructure along the walls of the fracture. The three tests that will be applied to all four

samples are shown in **Table 4.1**. While boundary conditions for each test will vary, however, we can maintain the material properties and interactions by ensuring that they all keep the same shared parameters, as listed in **Table 4.2**

Table 4.1 — Tests to Be Applied to All 4 Samples

Experiment Set	Boundary Condition	Occlusions?
1	Unconfined Tension (constant strain rate)	Not Present
2	Unconfined Tension (constant strain rate)	Present
3	Unconfined Compression (constant strain rate)	Present

Table 4.2 — Simulation Parameters for Unconfined Loading

Parameter	value
Number of Particles	50864
Strain Rate	100 <i>m/s</i>
Interaction Radius	1.5
Young's Modulus of Elasticity for Clay	11.6 <i>GPa</i>
Clay Tensile Fracture Stress	250 <i>kPa</i>
Clay Poisson Ratio	0.144
Clay Friction Angle	10°
Clay Density	2650 <i>kg/m³</i>
Young's Modulus of Elasticity for Carbonate	186 <i>GPa</i>
Carbonate Tensile Fracture Stress	4.1 <i>MPa</i>
Carbonate Poisson Ratio	0.17
Carbonate Friction Angle	34.3°
Carbonate Density	2160 <i>kg/m³</i>
Young's Modulus of Elasticity for Quartz	223 <i>GPa</i>
Quartz Tensile Fracture Stress	259 <i>MPa</i>
Quartz Poisson Ratio	0.11
Quartz Friction Angle	34.3°
Quartz Density	2650 <i>kg/m³</i>

4.2 Unconfined Tensile Fracturing in Occlusion-Free Materials

As a necessary point of reference, it is important to first determine the default stress-strain behavior and the microfracture shapes that evolve during uniaxial testing. In a material that is homogeneous and free from the effects of mineral occlusions, a reasonable expectation is that the presence and shape of kerogen pores or the random arrangement of particle packing would account for differences in the shape of a microfracture evolving during pure uniaxial tension. As stated in section 4.1, I will be performing each trial using the simulation parameters listed in **Table 4.2**, with the load applied in the horizontal direction (**Fig. 4.1**).

I note that while I could study the effect of mineral occlusions by substituting clay for all materials, sample C and D actually have clay as a minority component. Thus, for these single-material simulations, each particle in the simulation is defined as the dominant material for that sample (**Fig. 4.2**). For example, in Sample A, which holds clay as its most dominant material, I assign all particles with clay material properties.

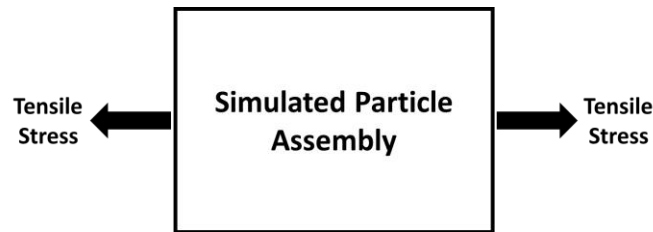


Figure 4.1 — During tensile testing, the edges of the DEM particle assembly is loaded in the horizontal direction. The tensile stress proceeds at a constant strain rate until a user-defined endpoint is encountered.

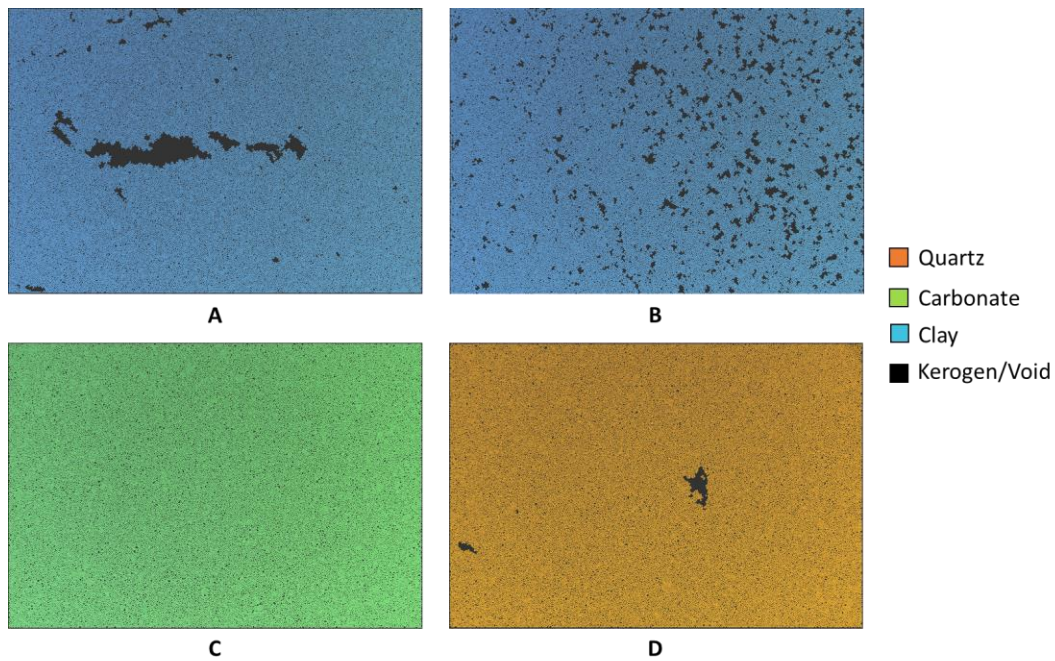


Figure 4.2 — Particle assembly from all four samples, based on SEM/EDS images. As a first test, all non-empty spaces are converted to the dominant material of that sample. For samples A and B, the dominant material is clay. For samples C and D, the dominant materials are carbonate and quartz, respectively.

In addition to the properties of the materials in the four shale samples, the specific packing of the particles in every DEM simulation will modify the calculated mechanical properties and fracture path. Because the particle growth method results in randomly distributed flaws in the packing of the particles, a randomized source of fracture generation is inherent to the particle assembly. Thus, instead of an idealized single fracture that forms perpendicular to the primary axis of stress, multiple fractures may evolve in a randomly distributed fashion until a dominant fracture that has completely cleaved through the material is generated.

Despite the inherent randomness of the particle packing, I can perform uniaxial tension on each of the four samples in **Fig. 4.2**. Because each sample will have a different shapes and volume fractions of pore space, one would expect very different fracture shapes if pores had a strong effect on fractures. Visualization of the fractures can be found in Fig. 4.3 to Fig. 4.6

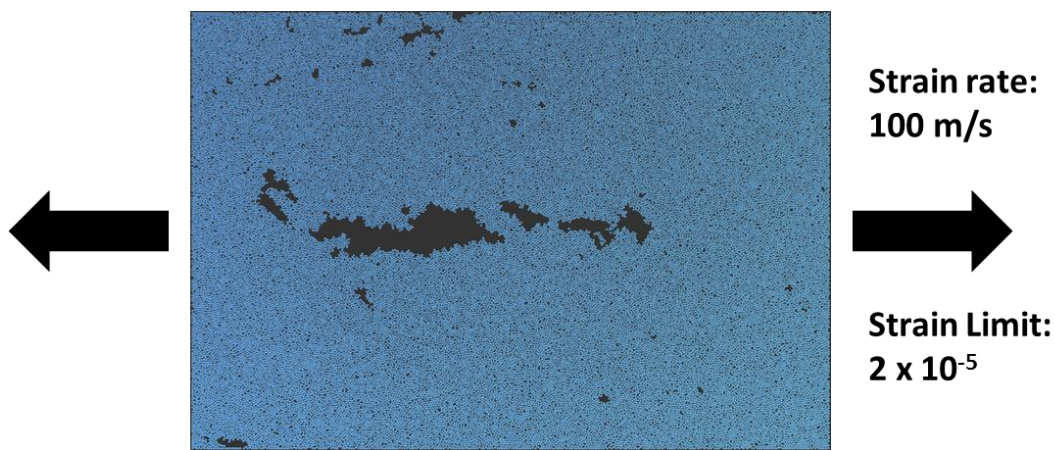


Figure 4.3 — A simulated region based on the structure of sample A. Tensile loading is applied to maintain a strain rate of 100 m/s. Each of the other three DEM particle assemblies shown in Fig. 4 will have also have a tensile force applied upon them under a constant strain rate of 100 m/s.

Tensile loading of Sample A shows that the relatively large pore in its center has a dramatic influence on fracture shapes in the sample (**Fig. 4.3**). While fracture shapes are not identical, it is apparent that the primary fracture paths are very similar to each other. As we can observe the outlined fracture zones, the fracture paths are quite similar in appearance despite the fact that the DEM particles are randomly packed. This is strong evidence of the powerful influence that pre-existing pores and effects have on fracture behavior.

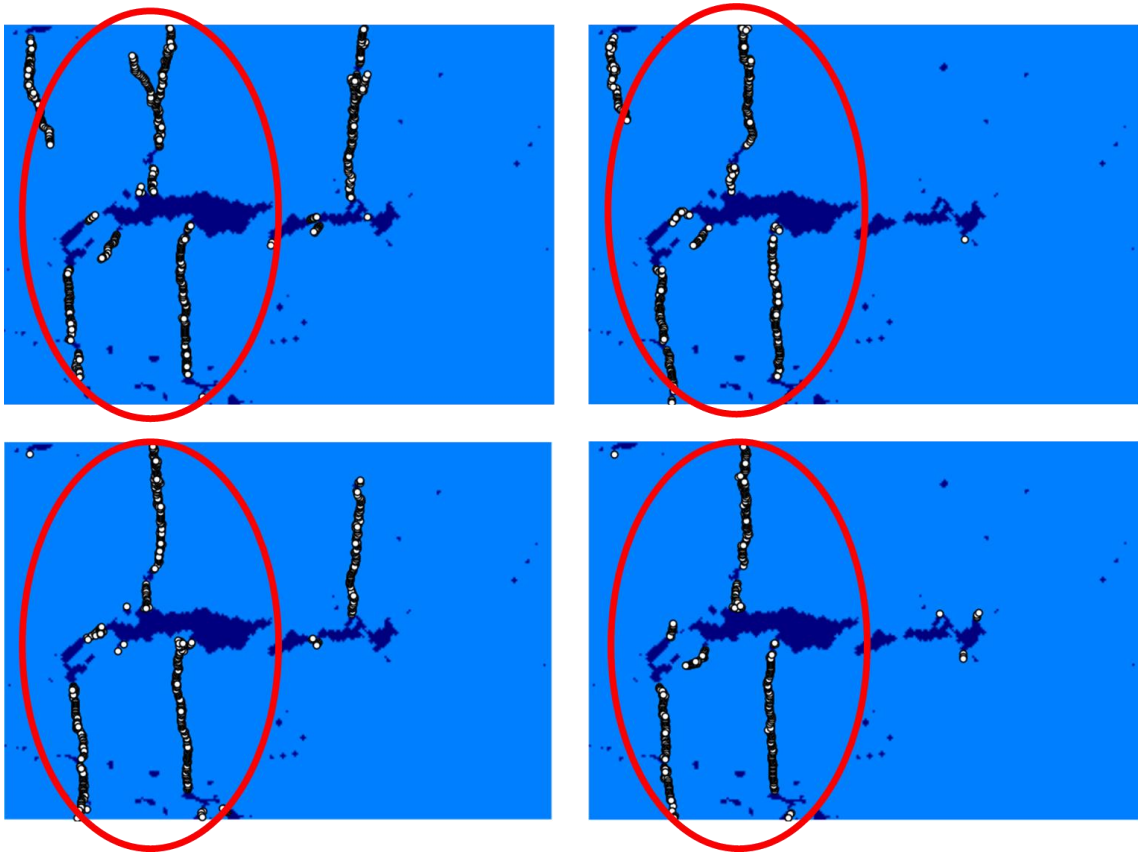


Figure 4.4 — Pure clay fracture patterns of sample A. The light blue section is defined as clay, the dark blue section is defined as pore space, and white dots represent bonds that have fractured during loading. The only difference between every trial is the packing of the particles themselves. Outlined in red is the fracture shape that is shared by all four representative samples.

Tensile loading of sample B shows that a distributed number of large pore can randomize the network, though certain general shapes appear to generally evolve (**Fig. 4.5**). In addition, as seen in **Table 4.3**, these distributed pores collectively weaken the macroscopic Young's modulus of elasticity and critical fracture stress far greater than a single large pore.

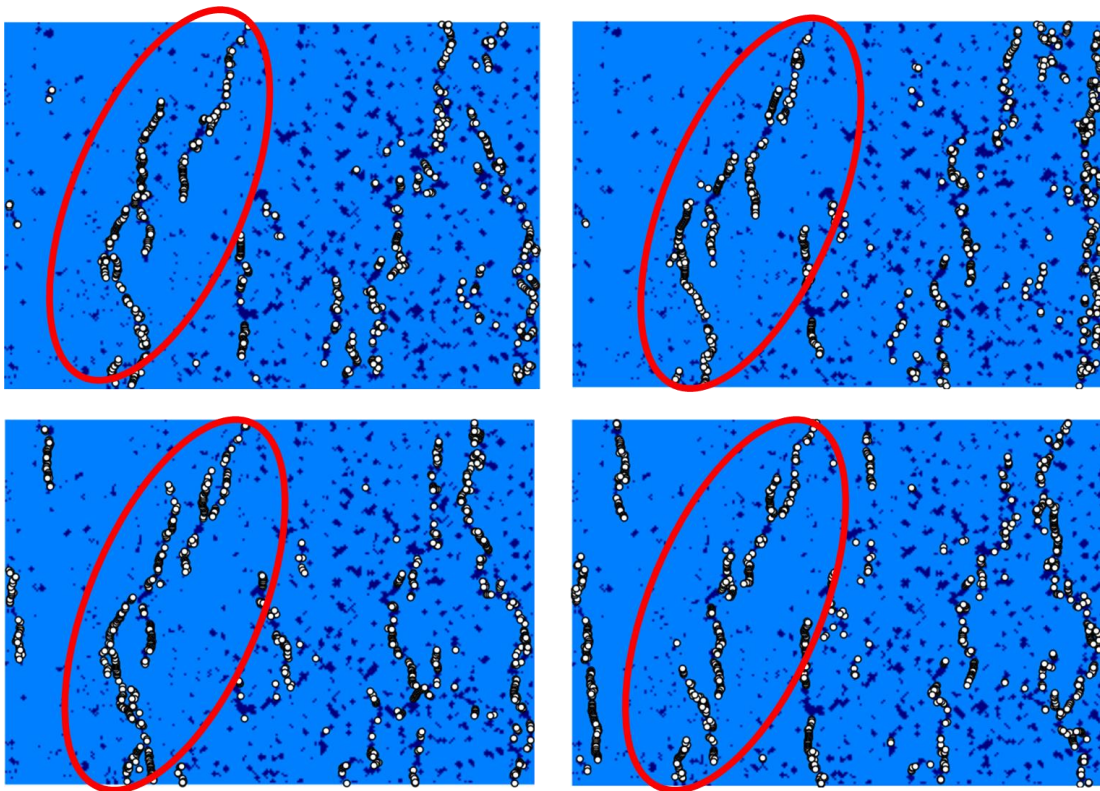


Figure 4.5 — Pure clay fracture patterns of sample B. In sample B, the light blue is clay, the dark blue section is pore space, and white dots represent bonds that fractured during tensile loading. The only difference between every trial is the packing of the particle assembly. In this case, because there are no overriding features in the mesh, no dominant fracture network has evolved yet.

Of the four samples, only sample C does not have any pores. This means that the fracture shapes observed in this sample are controlled only by the boundary conditions and the inherent randomness in the particle packing (**Fig. 4.5**). This is reflected in the far greater number of fractures in the simulated space as well as the elevated modulus shown in **Table 4.3**.

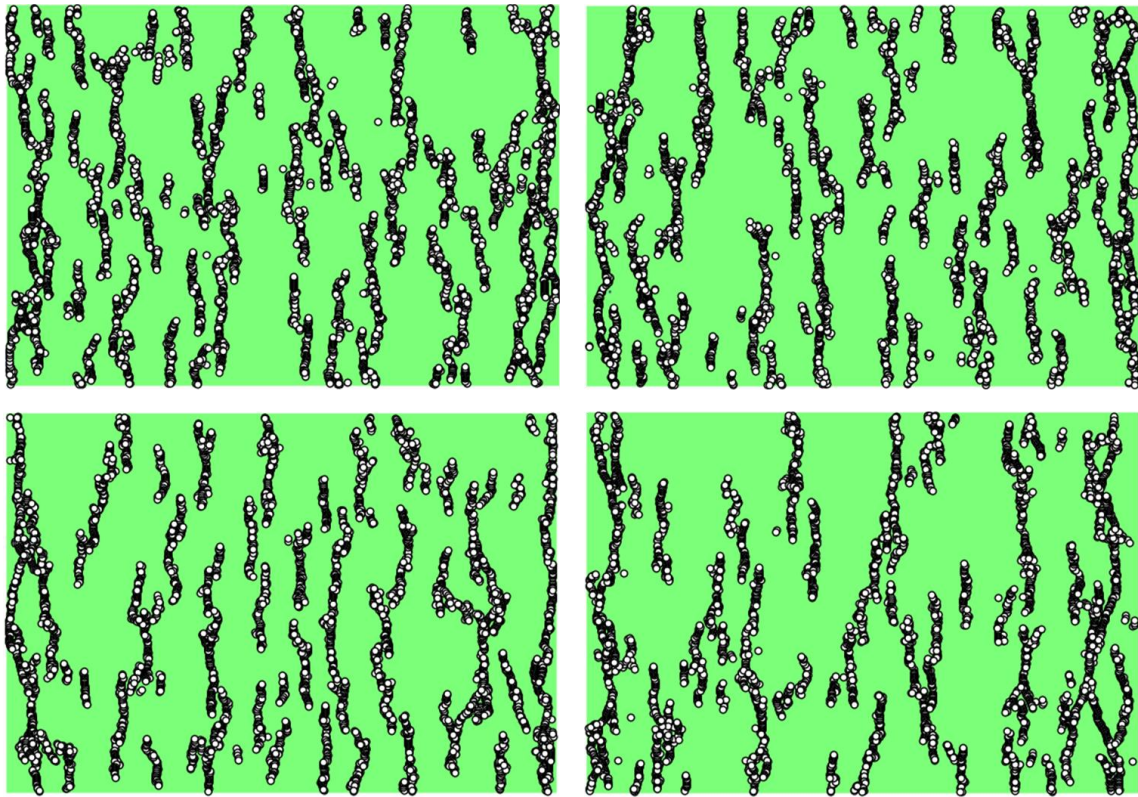


Figure 4.6 — Pure carbonate fracture patterns of sample C, the light green section is defined as carbonate and the white dots represent the position of bond fractures. While there is a general pattern to fracture vertically, I cannot observe predictable fracture location. Sample C contains no empty pores, so it is only the random packing of sample that results in these differences.

Similar to sample A, there is a central pore in Sample D, which has been simulated to be composed of pure quartz (**Fig. 4.6**). Though this pore is much smaller, it is still able to influence fracture growth such that a tensile load will predictably induce growth through this pore. Pragmatically, this stress concentration also dramatically reduces the estimated fracture stress of this material compared to a defect-free quartz block, as shown in **Table 4.3**.

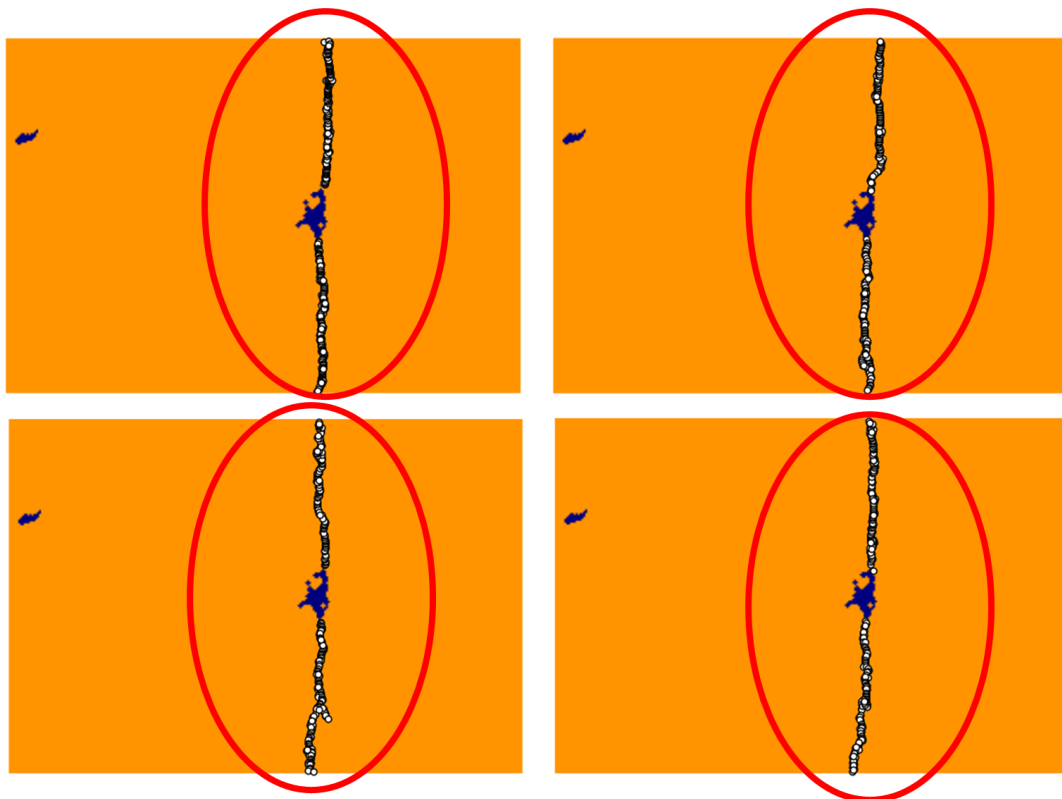


Figure 4.7 — Pure quartz fracture patterns of sample D. The orange represents quartz, the dark blue is once again pore space, and white dots represent bonds that have fractured. Above, I can see four very similar fracture patterns, suggesting the dominant role that a single flaw can have on a structure.

Table 4.3 — Material Properties Stresses (8 Trials per Sample)

Sample	Homogeneous Fracture Stress, <i>Pa</i> (Std. Dev.)	Homogeneous Mean Young's Modulus (Std. Dev.)
A	34.4 <i>kPa</i> ($\sigma=2.33$ <i>kPa</i>)	4.38 <i>GPa</i> ($\sigma=0.027$ <i>GPa</i>)
B	29.3 <i>kPa</i> ($\sigma=0.58$ <i>kPa</i>)	3.43 <i>GPa</i> ($\sigma=0.070$ <i>GPa</i>)
C	732 <i>kPa</i> ($\sigma=11.3$ <i>kPa</i>)	80.9 <i>GPa</i> ($\sigma=5.27$ <i>GPa</i>)
D	2200 <i>kPa</i> ($\sigma=99.2$ <i>kPa</i>)	86.3 <i>GPa</i> ($\sigma=6.89$ <i>GPa</i>)

4.3 Unconfined Tensile Fracturing in Shale Samples with Occlusions

I first attempted to determine the potential roles that occlusions may play in shale fractures by re-introducing these occlusions in my tensile stress simulations. I will use the same parameters shown in Table 4.2 and, as before, apply an unconfined tensile strain of 100 m/s to each of the four samples in this experiment, using the maps shown in **Fig. 3.4** to generate particle assemblies such as those shown in **Fig. 4.9** to **Fig. 4.12**. After including these occlusions in the simulation, the volume fraction of each of the major components of the four samples can be quantitatively calculated. The results of this effort are tabulated in **Table 4.4**.

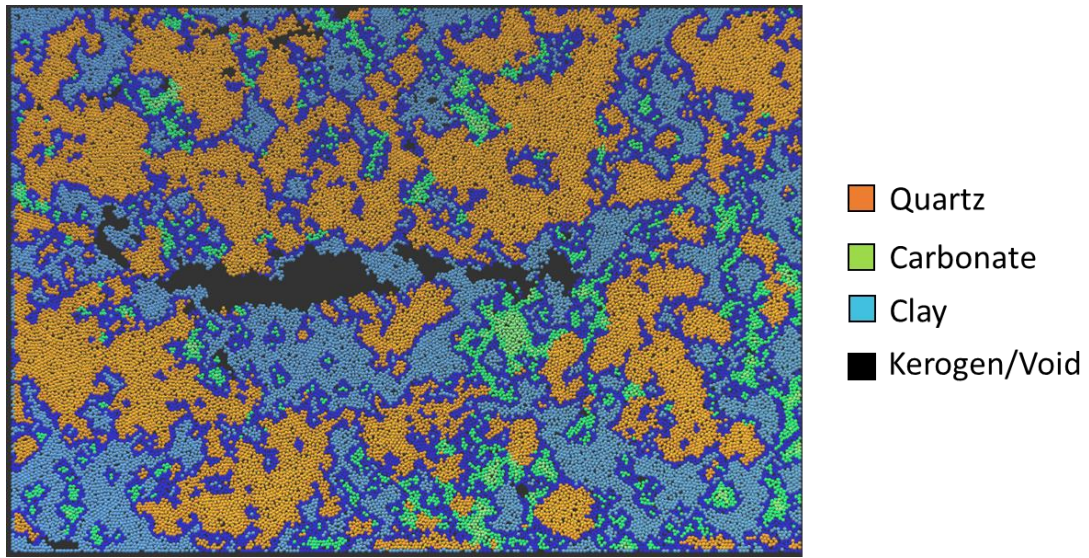


Figure 4.8 — Sample A DEM particle assembly, made from the material maps in **Fig. 3.4**

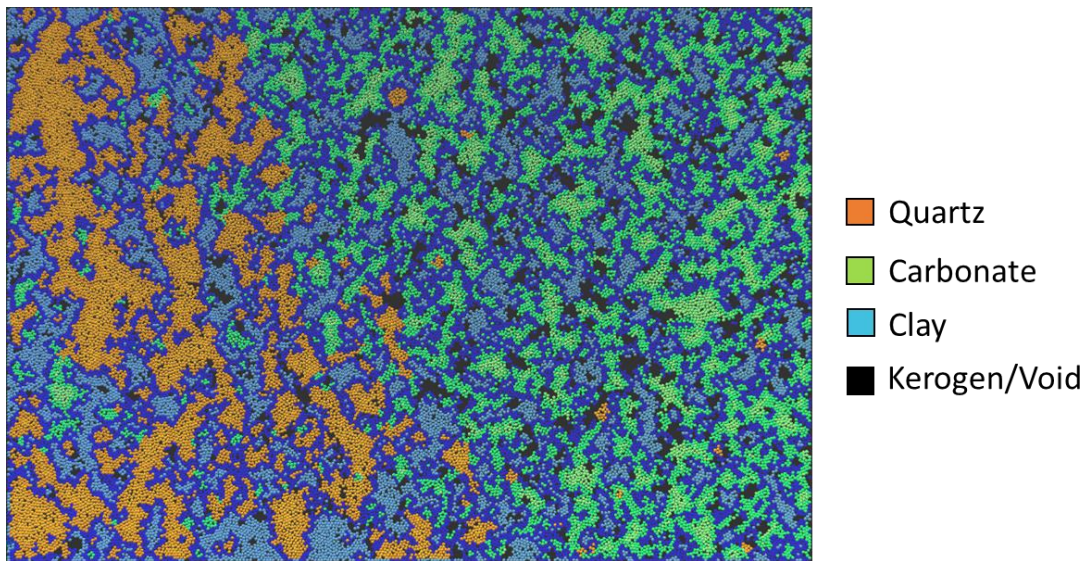


Figure 4.9 — Sample B DEM particle assembly, made from the material maps in **Fig. 3.4**

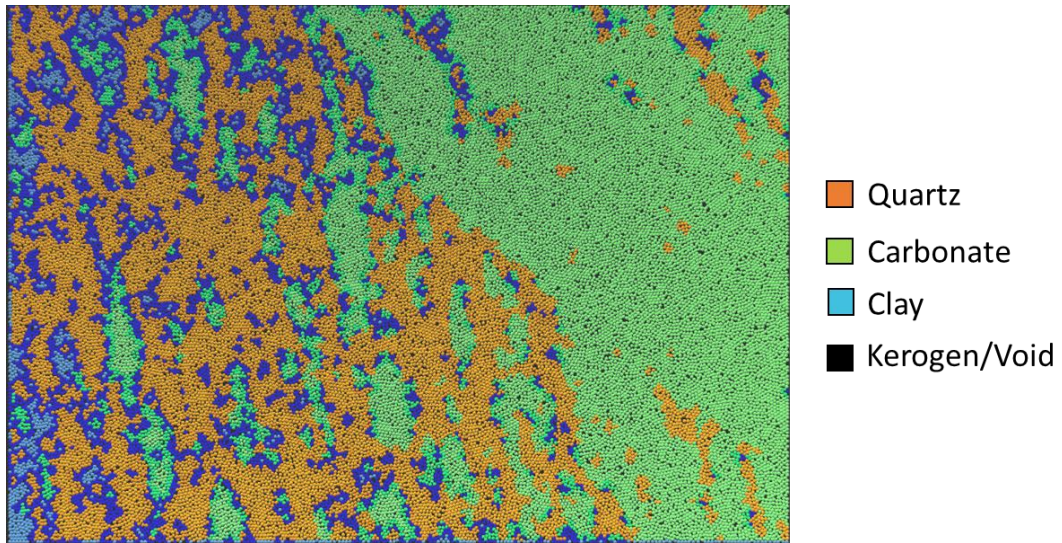


Figure 4.10 — Sample C DEM particle assembly, made from the material maps in **Fig. 3.4**

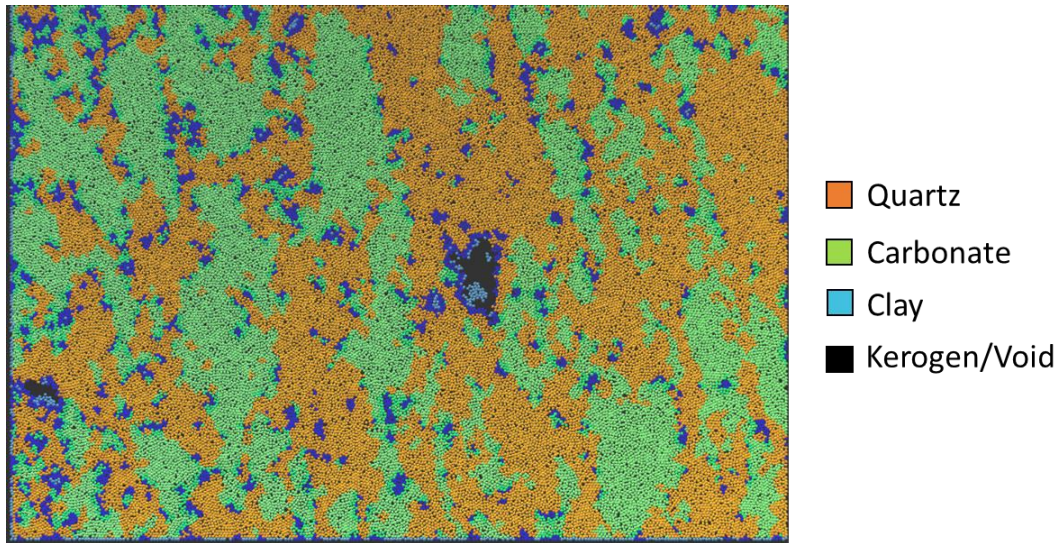


Figure 4.11 — Sample D DEM particle assembly, made from the material maps in **Fig. 3.4**

Table 4.4 — Sample Composition

Sample	Clay Percentage	Carbonate Percentage	Quartz Percentage	Kerogen/Void Percentage
1	50.4%	24.3%	18.1%	7.2%
2	50.0%	24.4%	18.3%	7.3%
3	17.1%	44.8%	38.1%	0.0%
4	7.5%	42.1%	49.9%	0.6%

While I am limited to analyzing only four shale samples, it is fortunate that the similarities in composition between Samples A and B makes it possible to glean significant insights into the relationship in mechanical behavior due to differences in their microstructure. Because the composition is so similar within these pairs, any measured differences in mechanical behavior between Samples A and B or Samples C and D is likely a result of the structural differences between the samples.

Due to the complexity of the materials to be deformed, the stress-strain plots will not be linear. This means that a specific method to estimate the Young's modulus of elasticity must be chosen for this set of experiments. Of the various methods to estimate the Young's modulus of elasticity, I selected the ASTM Tan_i method, which is to determine the modulus from the tangent of the stress-strain plot at the relative stress value i . This method was recommended as being the most consistent method of elastic-plastic materials (Santi, Holschen, and Stephenson 2000). However, while many use this method to find the tangent slope at 50% of the maximum stress experienced by the material, I instead measure the elastic moduli at 2% of the maximum stress in order to minimize the additional concern of heterogeneity-induced changes in ductile behavior.

As shown in **Fig. 4.12**, the mechanical response of Sample A shows a linear relationship between stress and strain during early deformation, behavior that holds true for all of the other three other samples as well. As the simulated volume continues to be strained, the bonds between particles reach their critical stress and fracture and the effective elastic modulus of the entire volume appears to be decreasing. After reaching a critical stress, enough bonds will be broken and the total stress can no longer increase. At this stress, the macroscopic interpretation of the mechanical response would be that the DEM particle assembly has completely transitioned into a plastic regime and the amount of stress required for further deformation will either remain approximately the same or decrease.

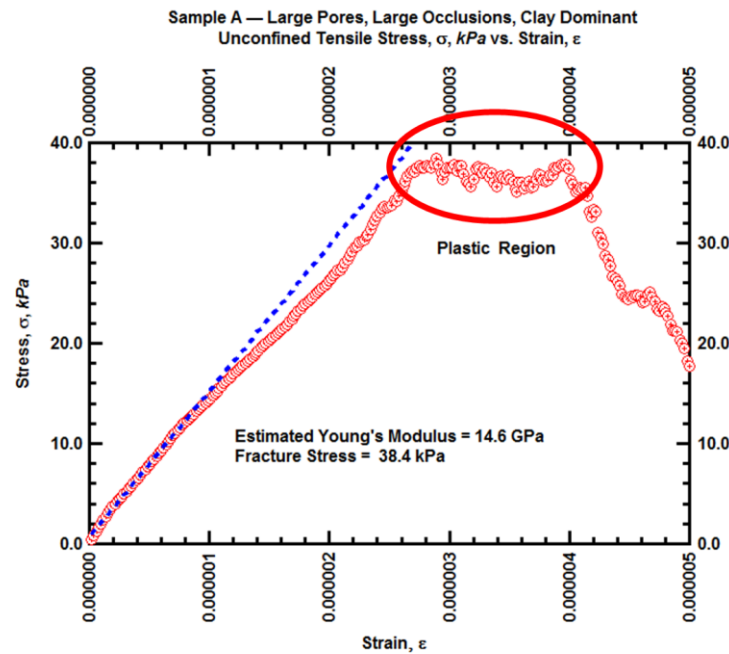


Figure 4.12 — Representative Unconfined Stress vs. Strain plot for sample A. The Tan_i method is used to determine modulus and represented by the blue dotted line. The region where a plastic regime is observed is shown by the red oval.

As can be observed in **Fig. 4.13**, Sample B has a much lower Young's modulus compared to Sample A. This may be due to the structure of Sample B, in which pores are distributed throughout particle assembly and the minerals much smaller and aligned perpendicularly to the clay occlusion (**Fig. 4.10**). As can be seen from **Table. 4.5**, this comparison is not unique to this single trial. The structural differences in Sample B results in a significantly reduced mean fracture stress and Young's modulus of elasticity (i.e. a modulus of 9.76 *GPa* compared to 13.9 *GPa*).

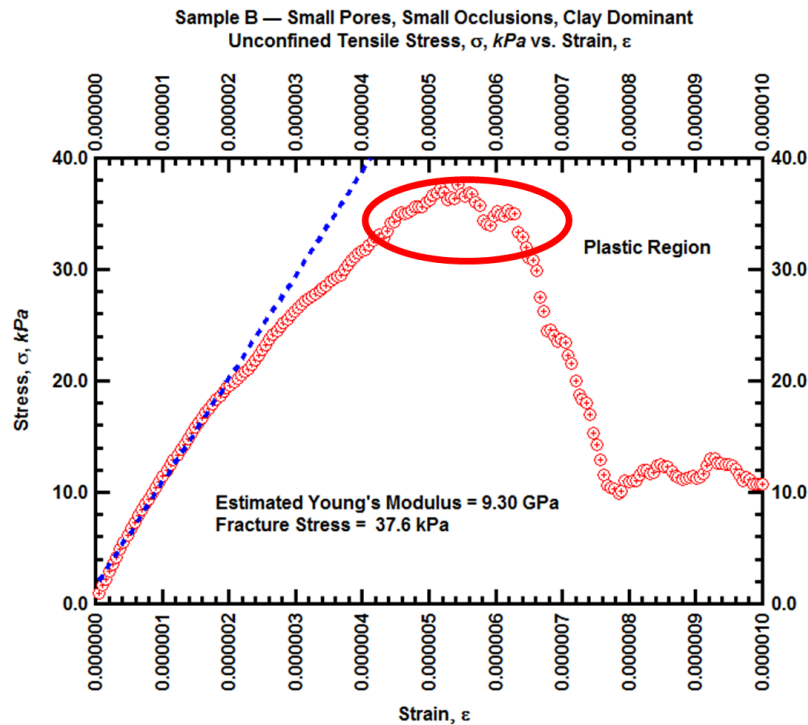


Figure 4.13 — Representative Unconfined Stress vs. Strain plot for sample B. The Tan_i method is used to determine modulus and represented by the blue dotted line. The region where a plastic regime is observed is shown by the red oval.

As observable in **Fig. 4.15**, Sample C has a much greater modulus and fracture limit compared to Sample A or Sample B. This is likely because the dominant materials in Sample C are carbonate and quartz minerals (**Fig. 4.11**). However, though the material itself is stronger, the general mechanical behavior appears the same.

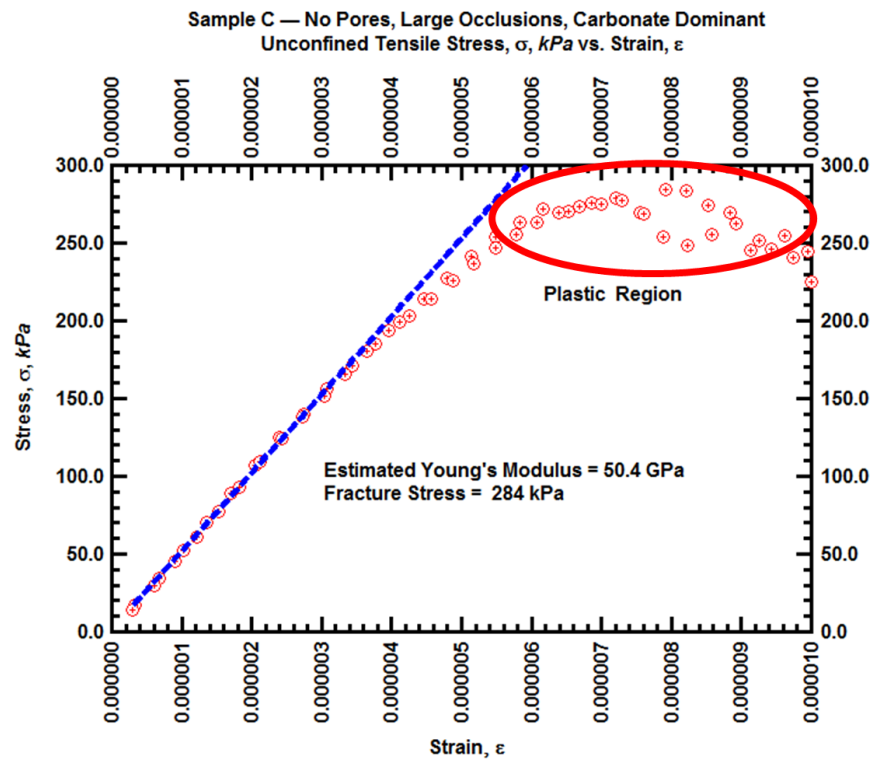


Figure 4.14 — Representative Unconfined Stress vs. Strain plot for sample C. The Tan_i method is used to determine Young's modulus and represented by the blue dotted line. Compared to Sample A and B, the modulus and fracture stress is significantly higher, likely because the dominant materials in this sample are carbonate and quartz, and also because there are no pores whatsoever in Sample C.

As observable in **Fig. 4.15**, Sample D has an even much greater modulus and fracture limit compared to Sample C. This is likely because the elevated volume of quartz in Sample D requires a greater amount quartz bonds to actually be stressed and fractured (**Fig. 4.12**).

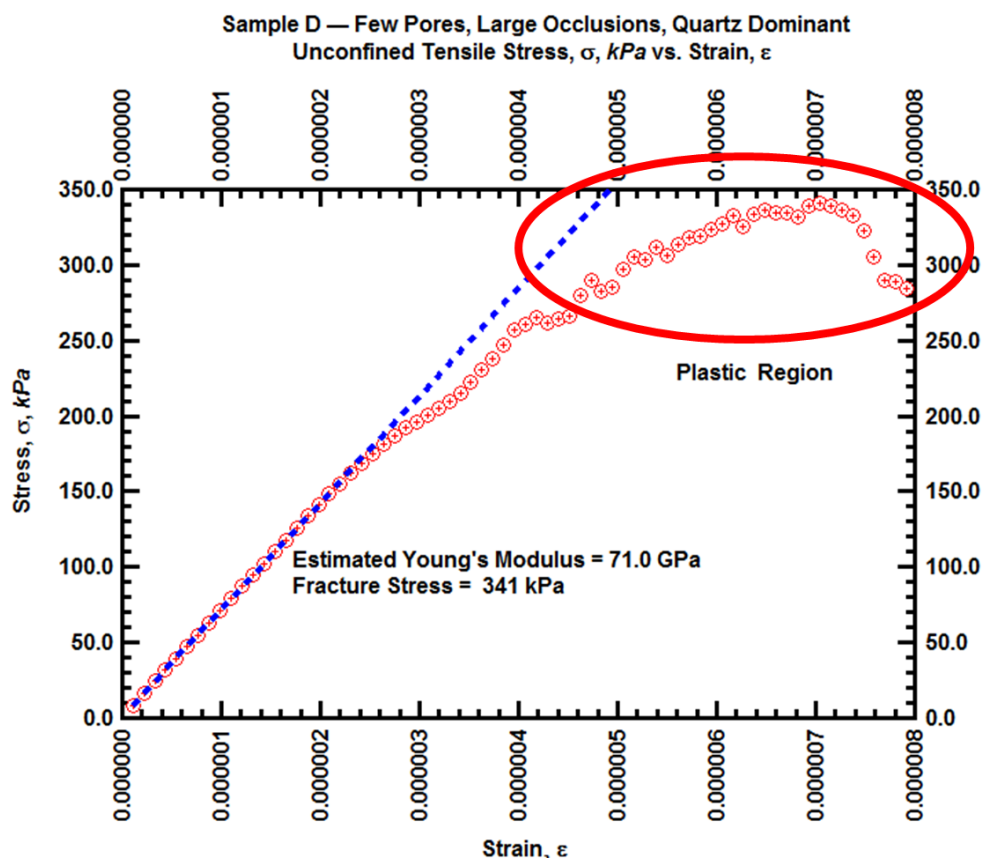


Figure 4.15 — Representative Unconfined Stress vs. Strain plot for sample D. The $\tan \epsilon$ method is used to determine Young's modulus and represented by the blue dotted line. Compared to Sample A and B, the modulus and fracture stress is significantly higher, likely because the dominant materials in this sample are carbonate and quartz, and also because there are no pores whatsoever in Sample C.

The mean Young's modulus of elasticity and the mean fracture stress for Samples A to D, as well as the standard deviations computed from the 8 trials (realizations) per sample, are shown in **Table 4.5**, which includes the corresponding values of clay as the reference material.

Table 4.5 — Young's Modulus and Fracture Stress under Tension (8 trials per sample)

Sample	Fracture Stress, <i>kPa</i> (Std. Dev.)	Modulus, <i>GPa</i> (Std. Dev.)
A	83.7 ($\sigma=7.55$)	13.9 ($\sigma=1.28$)
B	75.6 ($\sigma=3.23$)	9.76 ($\sigma=0.364$)
C	275 ($\sigma=17.17$)	51.0 ($\sigma=1.99$)
D	357 ($\sigma=22.9$)	70.2 ($\sigma=1.89$)
Clay (From Calibration Experiments)	96 ($\sigma=3.2$)	5.0 ($\sigma=0.32$)

It is apparent that the inclusion of minerals in the shale microstructure does not appear to strengthen the material in any linear fashion. For example, though the mineral occlusions replaced 42.7% of the area in sample B, fracture stress only increased by 7.85%. This strengthening effect be even less pronounced 3D, due to the extra degree of freedom allow for fractures to propagate around obstacles. On the other hand, when seen from the opposite perspective of the effect of clay addition to a “harder” homogeneous mineral, it can be stated that the presence of clay at the 17.1% and 7.5% levels in Samples C and D, respectively, lead to decreases by orders of magnitude in the fracture stress of the shales.

These results tend to indicate that fracturing is much more influenced by the weakest element than by the strongest element of the shale structure.

What is also clear is that while shale composition may have a first-order effect on its geomechanical behavior, shale microstructure may also play a significant role on the material strength, as evidenced by the comparison of the mechanical properties of Samples A and B. Although both samples are nearly identical in terms of their macroscopic compositions, they are drastically different in terms of their microstructure. Comparing **Fig. 4.16** and **Fig. 4.17**, it can be observed that Sample A is comprised primarily of large quartz occlusions with two relatively large pores at the center, while sample B has a far more disperse arrangement of mineral occlusions. Because the simulation tests involved tensile stress application in the horizontal direction, the aligned quartz particles in Sample A effectively generate a far greater elastic modulus and fracture stress while also inhibiting previously growth of fractures beneath the large central pore (**Fig. 4.16**).

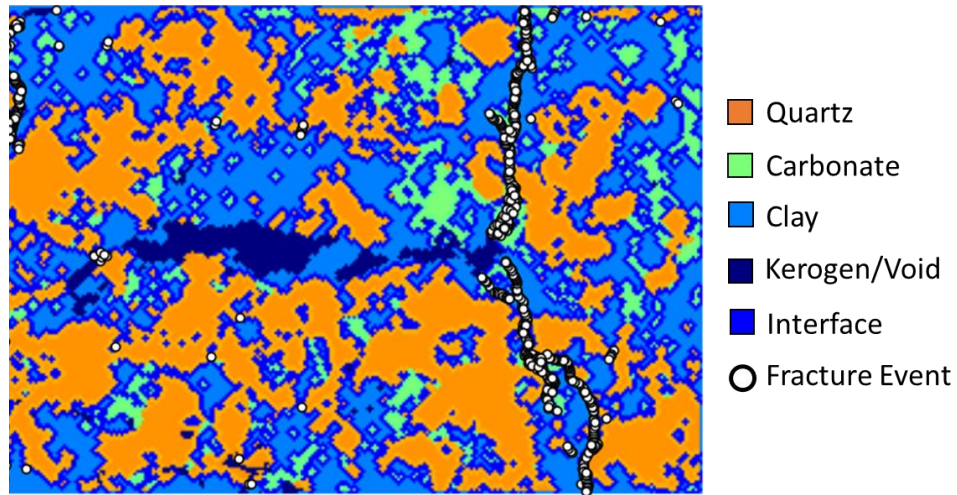


Figure 4.16 — Example of fractures that evolve under tension for Sample A. Unlike the fractures for a pure-clay material in **Fig. 4.5**, this fracture does not come in contact with the central pore because this pore is now surrounded by quartz occlusions that inhibit fracture growth.

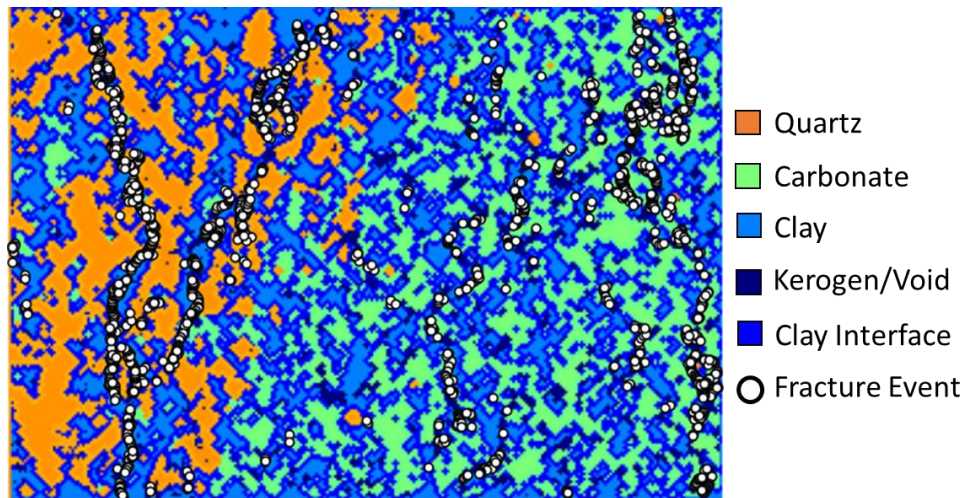


Figure 4.17 — Example of fractures that evolve under tension for Sample B. Unlike sample A, the distribution of mineral occlusions do not completely isolate any pore or significantly impede vertical fracture growth. Thus, these occlusions are less effective at strengthening the sample.

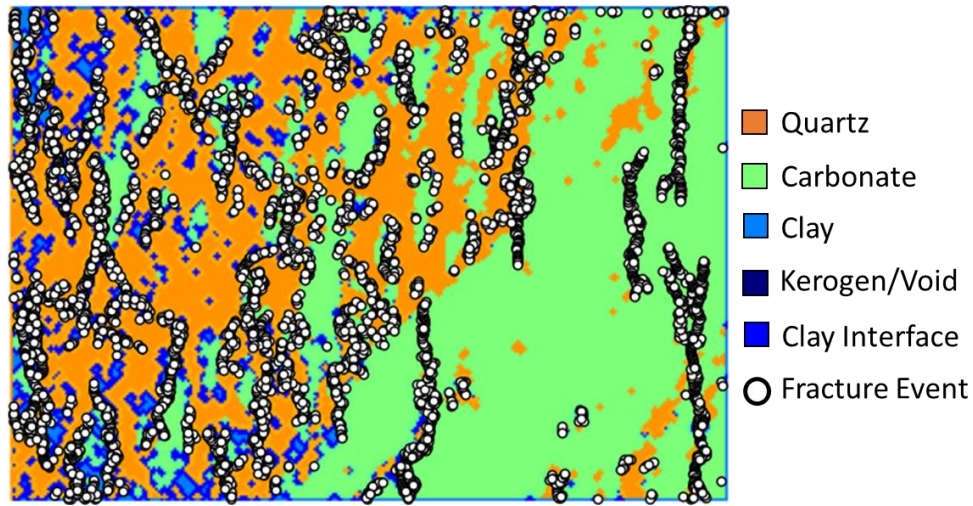


Figure 4.18 — Example of fractures that evolve under tension for sample C. Without a significant quantity of clay, fracturing must occur within the minerals. A visual inspection of these fractures show that the majority virtually all occur at or near the boundary interfaces between quartz and carbonate.

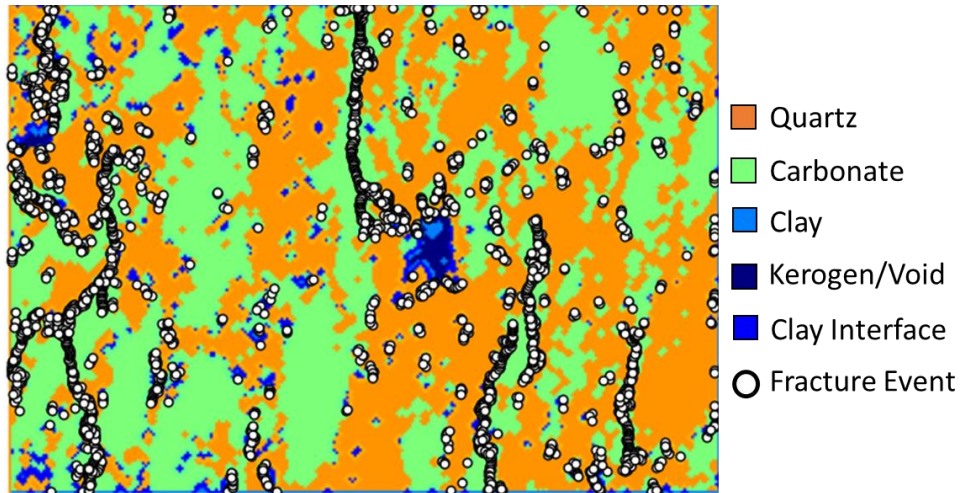


Figure 4.19 — Example of fractures that evolve under tension for sample D. Though there is a central pore, the quartz occlusion surrounding the pore on the bottom arrests fracture growth and induces fractures to grow at edges of the quartz occlusions instead.

To better understand and quantify the effect of occlusions on the fracture location of initiation and geometry, I investigated the relationship between the shale microstructure

and the number of fractures generated in the clay, the mineral occlusions, and the boundary interface between different materials. Though individual fracture patterns may vary between different realizations (trials) of the sample due to the randomness of particle packing, we would expect that the ratio of fractures occurring within and outside of specified regions to be consistent.

Specifically, I defined V_{ITZ} as the fraction of the area occupied by an interface relative to the total simulated volume. This area is generated by creating a virtual interface material that is approximately one particle wide and is located along the boundaries between (and separating the) different materials. Defining ϕ_{ITZ} as the ratio of fractures that involved at least one interface particle relative to the total number of recorded fractures, ϕ_{ITZ} would be approximately equal to be V_{ITZ} if there were no biases to fracturing near the interface of materials. The simulations showed that this was not the case, as fractures were strongly biased towards occurring along the interface between materials regardless of the dominant mineral of the sample (**Fig. 4.16** to **Fig. 4.19**). The majority of fractures appeared to occur at the interface between different materials, suggesting a clear relationship between fracture pattern and the interface of a mineral occlusion. This behavior was observed across all four samples at 2×10^{-5} strain, as shown in **Table 4.6**.

Table 4.6 — Fracture-Interface Correlation at 2×10^{-5} strain (8 trials per sample)

Sample	V_{ITZ}	Φ (Std. Dev.)
1	55.4%	88.3% ($\sigma = 0.28\%$)
2	31.1%	83.2% ($\sigma = 0.37\%$)
3	39.3%	89.4% ($\sigma = 0.35\%$)
4	25.6%	82.8% ($\sigma = 0.37\%$)

It is important to recognize that no additional manipulation was performed to modify the strength of the bond between different materials to reflect any additional chemical interaction. By default, bonds between two different materials will adopt the weaker properties of the two materials. As an example, for a bond between a clay particle and quartz particle, the critical stress limit before the bond fractures will be stress limit of the clay particle. Thus, the fact that virtually all particles fracture on the interface particles is a reflection of stress concentrations occurring at these boundaries. Thus, based on bias towards interfaces shown in **Table 4.6**, it is very likely that microstructures can in a certain sense control fracture behavior by leading fractures along the interface.

4.4 Unconfined Compressive Fracturing

While unconfined tensile tests may reveal a great deal about the material properties of shales as they break, shales may also experience a compressive stress instead of a tensile stress during a hydraulic fracture treatment. The importance of this consideration led to the investigation of the role of shale microstructure on compression. As before, the two key questions in this scenario were whether the microstructure of different samples can result in different macroscopic quantities, and what the fractures look like. As before, the strain rate will remain at 100 m/s and all parameters will remain at the same value shown in **Table 4.2**. From a practical perspective, the only difference in the system is to ensure that compressive loading is applied instead of tensile loading.

It is well-known that compressive stresses are far greater than tensile stresses, with the ratio of the ultimate compressive stress (UCS) to the ultimate tensile strength (UTS) reported as approximately double to greater than 10 (Koncagül and Santi 1999, Zhou, Zeng, and Liu 2010). This behavior is inevitable in my current DEM simulations because particle bonds can only be broken through shear or strain under the simulation physics chosen for this project with few mechanisms to relieve stress in other ways. This is a known effect in DEM simulations, and compensating for this effect requires additional modifications and calibration methods that were not implemented in this project (Wang and Tonon 2010). However, though the mechanical properties of this DEM simulation will not be quantifiably correct, these results may still provide insight into the qualitative aspects of a microstructure's effect on fracturing. Representative stress-strain plots for each of the four samples can be seen in **Fig. 4.20** to **Fig. 4.23**.

As with the tensile fracture tests, I estimated the Young's modulus of elasticity using the Tan_{50} method. The advantage and disadvantage of this decision is that, while the elasticity estimates are more susceptible to minor changes at the beginning, they are less susceptible to inadvertently at capturing effects that occur long after plastic effects become important.

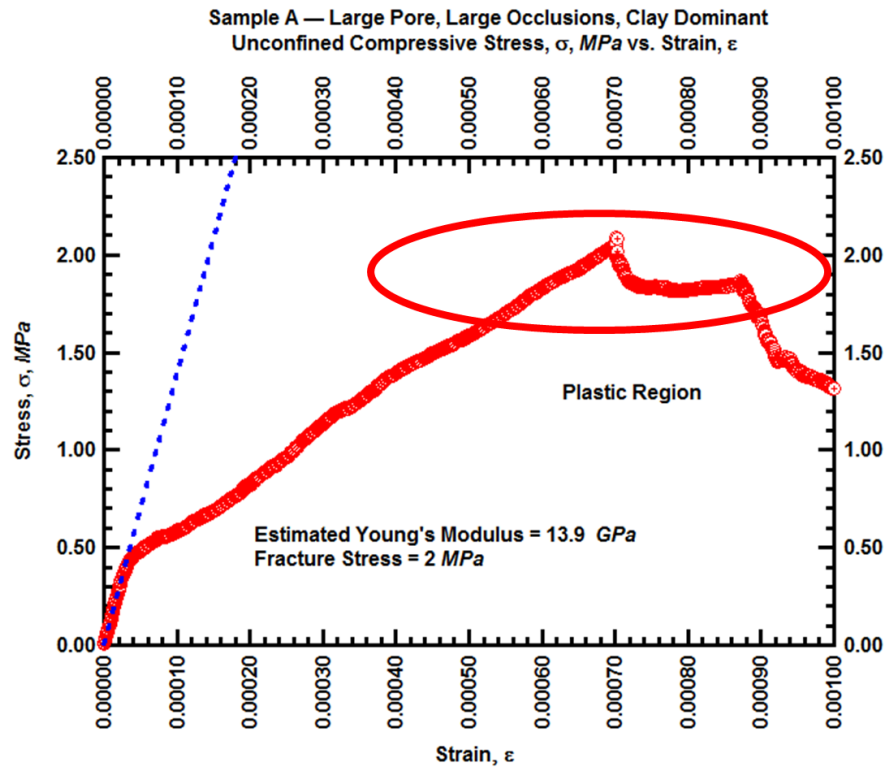


Figure 4.20 — Representative Confined Stress vs. Strain plot for sample A. In the first, uniaxial loading meets an elastic response from the material. However, enough stress is eventually applied that the material begins splitting across the central large pore (see **Fig. 3.9** as a reminder of this pore's size relative to the geometry). Thus, even before reaching the critical fracture stress, this division results a weaker overall particle system.

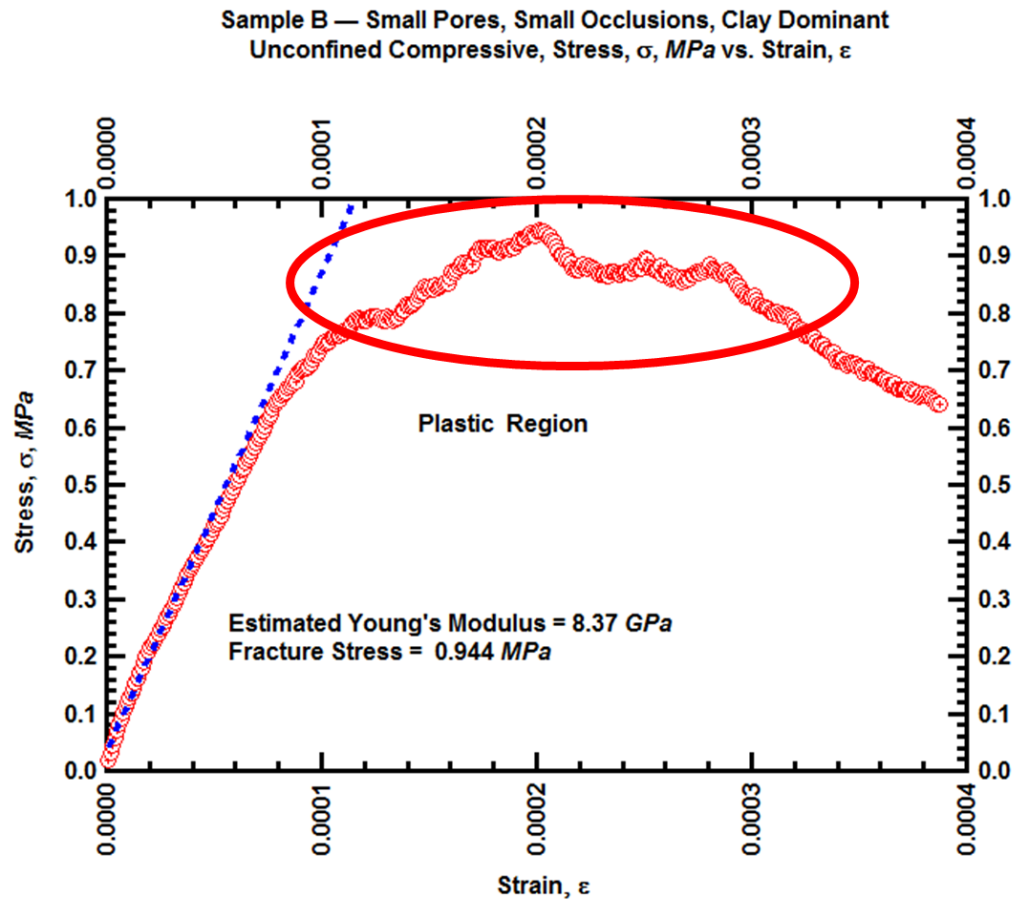


Figure 4.21 — Representative Confined Stress vs. Strain plot for sample B. Sample B shows a stress-strain response to compression in a qualitatively similar way to Sample A. In the case of sample B, however, the arrangement of the pores and occlusion are much more accommodating to shear failure due to the distributed nature of the occlusions (see **Fig. 4.10**).

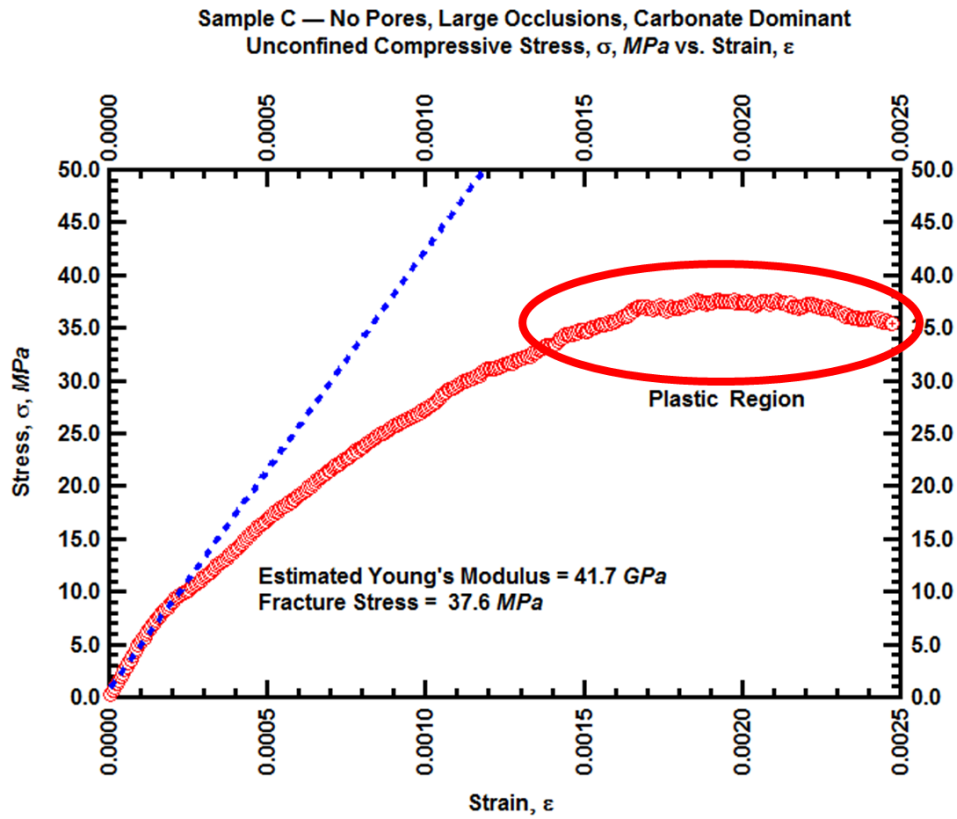


Figure 4.22 — Representative Confined Stress vs. Strain plot for sample C. Sample C shows a significantly greater modulus due to its elevated percent volume of quartz and carbonate greater mineral count. We note that its estimated initial Young's modulus of elasticity moduli is actually less than the one estimated from tensile testing.

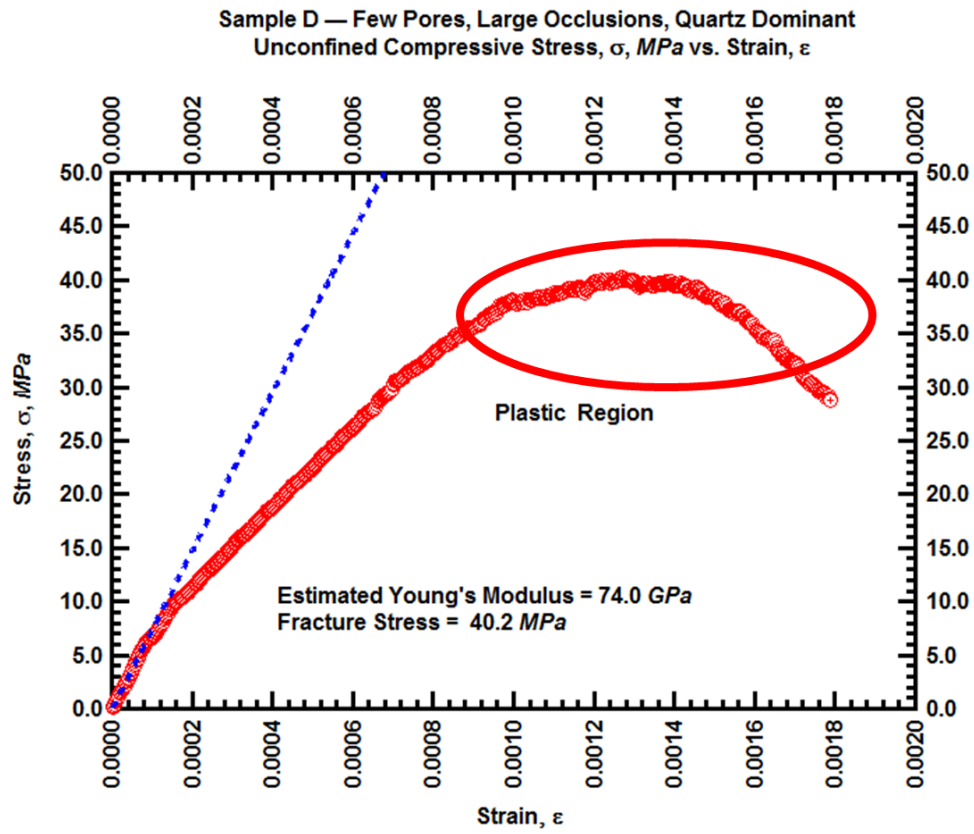


Figure 4.23 — Representative Confined Stress vs. Strain plot for sample D. As expected, sample D shows the greatest Young's modulus and fracture stress due to its relatively high quartz fraction. Though there are some pores in the volume, they amount to less than 6%.

To provide an estimate of the material properties, each sample was run under compression 8 times to produce plots similar to those of **Fig. 4.20** to **Fig. 4.23** and the summary of these trials is provided in **Table 4.7**. For comparison purposes, the estimated Young's modulus from the tensile tests are also included. What is found is that the two Young's moduli estimates from these two moduli are quite similar. Logically, this should be the case, as I minimize the opportunity for bond fractures to change the mechanical response, while the physical model I implemented makes no distinction between tensile and compressive stress.

Table 4.7 — Material Properties Estimate From Compression (8 Trials per Sample)

Sample	Tensile Young's Modulus, <i>GPa</i> (Std. Dev.)	Compressive Young's Modulus, <i>GPa</i> (Std. Dev.)	Compressive Fracture Stress, <i>MPa</i> (Std. Dev.)
A	13.9 ($\sigma=1.28$)	14.6 ($\sigma=0.106$)	3.19 ($\sigma=0.208$)
B	9.76 ($\sigma=0.364$)	8.6 ($\sigma=0.449$)	0.950 ($\sigma=0.0653$)
C	51.0 ($\sigma=1.99$)	41.7 ($\sigma=1.29$)	39.8 ($\sigma=3.58$)
D	70.2 ($\sigma=1.89$)	73.2 ($\sigma=2.54$)	40.8 ($\sigma=3.96$)

The difference between compressive and tensile stress extends to fracture propagation as well. A representative visualization of fractures in each of the four samples can be found in **Fig. 4.24** to **Fig. 4.27**.

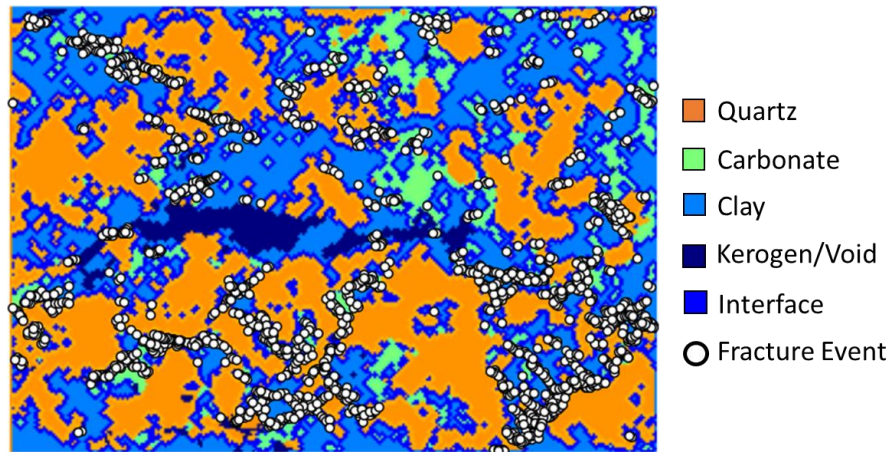


Figure 4.24 — Example of fractures that evolve under tension for sample A. As can be seen from Sample A under compression, compressive fractures are very different from tensile fractures. However, even in compression, fractures appear to be clustered around the interfaces of different materials and propagate from existing fractures.

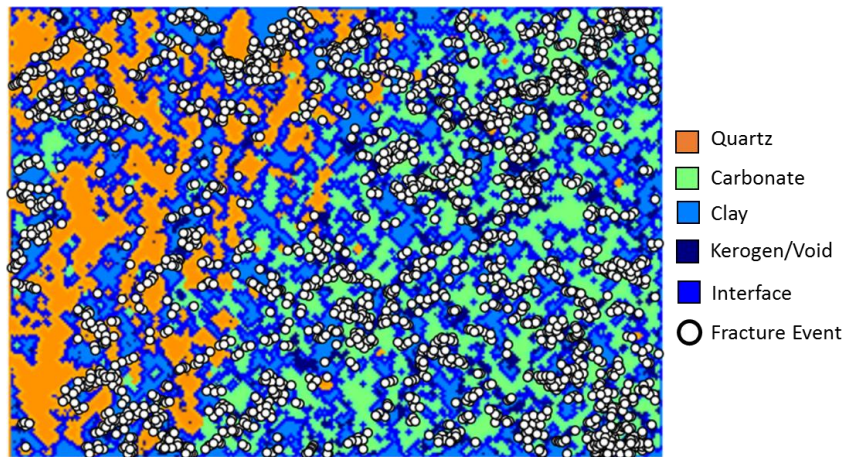


Figure 4.25 — Example of fractures that evolve under tension for sample B. The general trend of fractures to occur at the interface between clay and mineral occlusion is also evident in the Sample B. One possible reason that the compressive stress limit is so much greater in Sample A is that, unlike the quartz in Sample A, the quartz occlusions in Sample B are relatively small and thus the clay mortar can continue to be primary mineral to be fractured without being ‘trapped’ by quartz.

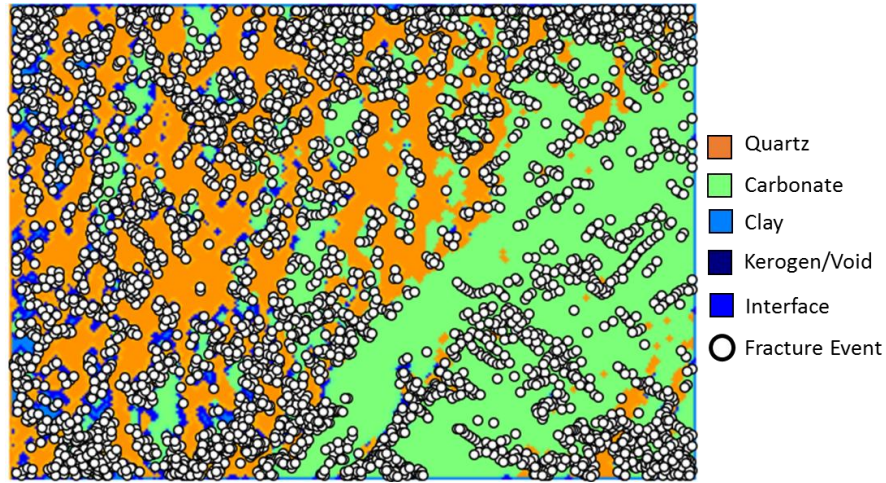


Figure 4.26 — Example of fractures that evolve under tension for sample C. As can be seen from this image, the heavy quartz presence and lack of clay ensures that some degree of fracturing occurs in the carbonate component. Of course, this dramatically increases the stress needed to fracture the simulated material.

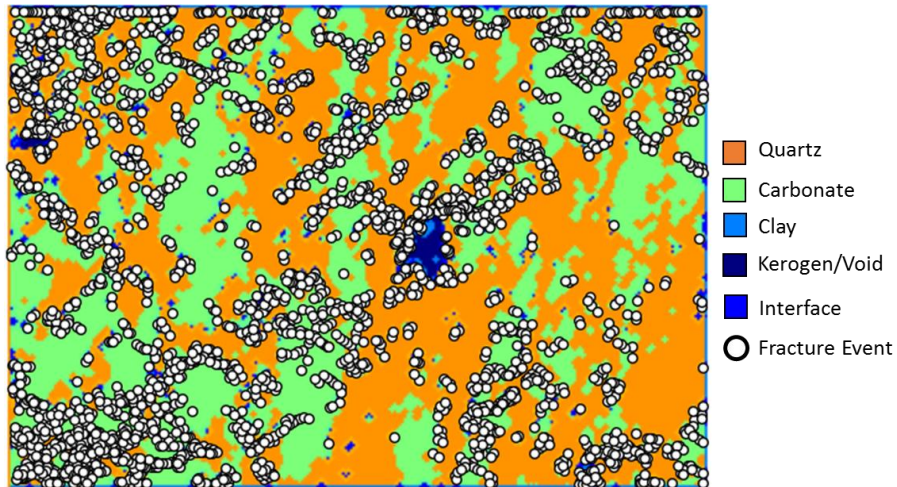


Figure 4.27 — Example of fractures that evolve under tension for sample D. In Sample D, compression may induce fracture growth around the pore space, especially in carbonate. One may wonder why a similar effect is not noticed in Sample A (**Fig. 4.24**), though this may have to do with relative abundance of clay near the boundary which can already to deform.

Observing these fracture behavior in these four samples, the most striking difference these fractures exhibit when contrasted with the ones generated in section 4.3 is that these fractures are far more distributed throughout the mass. Unlike the tensile stress simulation, fractures appear to be made primarily through shear stress. This observation is further supported by the characteristic 45° fracture angles observable in all 4 samples but especially in **Fig. 4.26** and **Fig. 4.27**, loading will continue to increase until a sufficient number of particles have been sheared out of the simulated volume.

The behavior described above will effectively cause fractures to occur throughout all of the elements in a DEM simulation if the compressive stress cannot be released in any other way. However, at low-to-medium stresses before the fracture stress is achieved, fracturing under unconfined compression appears to be both controlled by the geometry of boundaries between materials in the shale microstructure. Perhaps the striking example of this effect is between sample A and sample B. As previously noted, the percent-by-composition of these two materials are extremely similar. However, the pores of sample B are far more evenly distributed and there are no large, continuous mineral occlusion which might completely surround a pore.

To quantify this effect, I repeated the same correlation between fracture and the interface region for unconfined compression, evaluated at the maximum stress, tabulating both the ratio ϕ when the measured stress is 25% of the mean maximum fracture stress, and also when the applied compressive load has reached the compressive-limit and can no longer increase. These results are compiled in **Table 4.8**. As we can observe, though

compression does result in a significantly greater number of non-interface bonds being fractured, interfaces are still disproportionate part of mechanical fracturing.

Table 4.8 — Fracture-Interface Correlation Under Compression (8 trials per sample)

Sample	V_{ITZ}	$\Phi_{ITZ-25\%}$	$\Phi_{ITZ-compressive-limit}$
A	55.4%	74.0% ($\sigma = 0.33\%$)	55.4% ($\sigma = 0.33\%$)
B	31.1%	85.6% ($\sigma = 0.57\%$)	88.68% ($\sigma = 0.9\%$)
C	38.02%	85.5% ($\sigma = 0.44\%$)	55.4% ($\sigma = 0.6\%$)
D	8.54%	47.2% ($\sigma = 0.36\%$)	37.6% ($\sigma = 0.39\%$)

In general, we would expect the information from the lower stress values to be more reflective of what might actually occur in any pragmatic hydraulic treatment. It would be unlikely that sufficient water pressure would be applied to effectively pulverize rock so they would break down into their constituent grains. Regardless, what we have is more quantitative evidence of what we can visually observe in **Fig. 4.24** to **Fig.4.27**: interfaces between different materials still serve as a source of stress concentration during fracturing.

4.5 Discussion

Based on the evidence provided by including and excluding mineral occlusions in tensile microfracturing tests, the presence of mineral occlusions is found to lead to a quantitative strengthening of the Young's modulus of elasticity and of shale fracture stress. Experiments accounting for and omitting occlusions in the simulations indicates that the weakening effect of a weak material on an otherwise strong material is far greater than the strengthening effect of a strong material on a weak material. Furthermore, the region most vulnerable to fracture-inducing stress appear to be the interfaces between the different materials. Thus, even without any special weakening or decreased elastic modulus or stress, fracturing is strongly biased to occur at the interface of different minerals under tension. The effect that these interfaces have on fracturing has been observed in experimental studies of fracturing in aggregate, brittle materials under tension (Nasseri and Mohanty 2008, Eberhardt, Stimpson, and Stead 1999, Mahabadi et al. 2012).

Even in compression, the microstructure strongly influences the fracture shapes by favoring fracturing at the interface of materials, and this effect is only reduced at stresses greater than the macroscopic failure stress of the sample shales I investigated. Perhaps the clearest example of this behavior is Sample A. With its relatively horizontal mineral occlusion and large pores and its fracture behavior when occlusions were not considered, one would reasonably expect that a fracture would evolve vertically through the center of the sample. However, the presence of a mineral occlusion mineral at the edges of the central pore inhibits fracture growth and thus a dominant fracture evolves in the adjacent clay region.

CHAPTER V

SUMMARY, CONCLUSION AND FUTURE WORK

5.1 Summary

In this thesis, we have proposed a method to integrate SEM/EDS data into a model that represents a shale matrix. This model is then used to create a numerical simulation study of the microscale fracture behavior of a synthetic shale sample where the properties of this synthetic sample are based on the Niobrara shale. While this method was explicitly designed for 2D image data, the methodology can also be adapted for 3D structures. The major task of this work was to study the behavior of induced microfractures found in heterogeneous shale microstructures.

Previous studies of microfracturing in shales using the discrete element method have focused on the treatment of shale as a homogeneous material and/or a material containing only macroscopic occlusions. While these past studies provide guidance for the behavior of macroscopic shale samples in certain scenarios, this focus of this study was to examine specific microstructures of the Niobrara shale using nanoscale properties in order to study the potential role of microstructures in shale as these features relate to induced fracturing.

Specific to this thesis, I developed a process to convert EDS/SEM data from four samples into a DEM particle assembly, and then applied this method to study the microfractures generated by simulating uniaxial tests on these samples. As part of these simulations, I performed unconfined tensile loading and unconfined compressive loading. As a result of this work, I demonstrated the role of the boundary interface between different materials in

guiding the shapes of the created fractures under both tensile and compressive conditions in microscale fracturing.

5.2 Conclusions

The effective permeability of the stimulated reservoir volume (SRV) caused by induced and natural fractures is an important quantity that can dramatically change the decision on whether or not to drill a well. Improving the predictions of effective permeability in fractured media requires a more fundamental understanding of the physical mechanisms and macroscopic fracture evolution that lead to fracture generation. In this 2D study I have shown that the microstructure of shale and the mineral occlusions within these microstructures can provide guidance as to how microscale apertures and interfaces can develop into a discretely fractured system.

While these studies cannot be directly translated into a quantitative relationship because the 2D simulations used in this work cannot accurately capture the 3D structure of shale, this fact does not preclude using 2D simulations to develop more insight into shale microstructure. Moreover, I believe that the methods that have been developed and described in this thesis can be extended to 3D studies of shale, provided that sufficient 3D structural data are available.

5.3 Recommendations, Comments, and Future Work

This work could be continued as follows:

1. 3D microstructure data should either be acquired (or simulated). The modification to existing code for applications to 3D systems will depend on whether data is being acquired or generated, but one approach would be create 3D data array instead of 2D array as a material map. Once a 3D map is generated, minor modification of the existing YADE-DEM simulation code should suffice to be able to accommodate 3D deformations. Alternatively, one could generate a function (*e.g.*, a fractal function for fractures) as a means of producing an entirely artificial material and then assign material IDs based on that function.
2. Mechanical fluid flow should be coupled to the simulator, either after post-processing or as a part of the simulation itself. This task would provide a proxy permeability estimate and could also be used to determine the pressure/treatment cycles required to dislodge an occlusion from a fracture wall.
3. More complex material models should be incorporated. At present, all of the materials considered are effectively being simulated as a linear elastic materials. One of the most important additions in this regard should be a more realistic assessment of the behavior of the kerogen. In this work, kerogen is treated as void space as a simplifying conditions. It is highly unlikely that kerogen would act as a void space when experiencing compression.

REFERENCES

- Akono, Ange-Therese, Pooyan Kabir. 2016. Microscopic fracture characterization of gas shale via scratch testing (in *Mechanics Research Communications*.
<http://www.sciencedirect.com/science/article/pii/S0093641315001822>.
- Ambrose, Jasmin. 2014. Failure of Anisotropic Shales under Triaxial Stress Conditions, Imperial College.
- Barbour, T. G., H. Y. Ko. 1979. Relationship Of Mechanical, Index, And Mineralogic Properties Of Coal Measure Rock. Proc., 20th U.S. Symposium on Rock Mechanics, Austin, TX.
- Bathija, Arpita Pal. 2009. Elastic Properties of Clays, Colorado School of Mines.
- Bennett, Kane C., Lucas A. Berla, William D. Nix et al. 2015. Instrumented nanoindentation and 3D mechanistic modeling of a shale at multiple scales (in *Acta Geotechnica* **10** (1): 1-14. <http://dx.doi.org/10.1007/s11440-014-0363-7>.
- Blair, S. C., N. G. W. Cook. 1998. Analysis of compressive fracture in rock using statistical techniques: Part II. Effect of microscale heterogeneity on macroscopic deformation (in *International Journal of Rock Mechanics and Mining Sciences* **35** (7): 849-861.
<http://www.sciencedirect.com/science/article/pii/S0148906298000096>.
- Bradski, Dr. Gary Rost, Adrian Kaehler. 2008. *Learning opencv, 1st edition*, O'Reilly Media, Inc. (Reprint).
- Brooks, Z., F. J. Ulm, H. H. Einstein. 2013. Environmental scanning electron microscopy (ESEM) and nanoindentation investigation of the crack tip process zone in marble (in *Acta Geotechnica* **8** (3): 223-245.
<http://dx.doi.org/10.1007/s11440-013-0213-z>.
- Chao, H Lin, Thomas E Parker. 1983. Tensile fracture strength of st cut quartz, DTIC Document.
- Chenu, C., J. Guérif. 1991. Mechanical Strength of Clay Minerals as Influenced by an Adsorbed Polysaccharide (in English). *Soil Science Society of America Journal* **55** (4): 1076-1080.
<http://dx.doi.org/10.2136/sssaj1991.03615995005500040030x>.
- Chong, KP, JL Chen, GF Dana et al. 1984. Indirect and direct tensile behavior of Devonian oil shales, Wyoming Univ., Laramie (USA). Dept. of Civil Engineering.

- Closmann, PJ, WB Bradley. 1979. The effect of temperature on tensile and compressive strengths and Young's modulus of oil shale (in *Society of Petroleum Engineers Journal* **19** (05): 301-312.
- Cundall, Peter A, Otto DL Strack. 1979. A discrete numerical model for granular assemblies (in *Geotechnique* **29** (1): 47-65.
- Curtis, Mark E, Carl H Sondergeld, Raymond J Ambrose et al. 2012. Microstructural investigation of gas shales in two and three dimensions using nanometer-scale resolution imaging (in *AAPG bulletin* **96** (4): 665-677.
- Duriez, J., L. Scholtès, F. V. Donzé. 2016. Micromechanics of wing crack propagation for different flaw properties (in *Engineering Fracture Mechanics* **153**: 378-398. <http://www.sciencedirect.com/science/article/pii/S0013794415007225>.
- Eberhardt, E., B. Stimpson, D. Stead. 1999. Effects of Grain Size on the Initiation and Propagation Thresholds of Stress-induced Brittle Fractures (in *Rock Mechanics and Rock Engineering* **32** (2): 81-99. <http://dx.doi.org/10.1007/s006030050026>.
- Grady, DE, RE Hollenbach. 1979. Dynamic fracture strength of rock (in *Geophysical Research Letters* **6** (2): 73-76.
- Greaves, George Neville, AL Greer, RS Lakes et al. 2011. Poisson's ratio and modern materials (in *Nature materials* **10** (11): 823-837.
- Gunsallus, K. L., F. H. Kulhawy. 1984. A comparative evaluation of rock strength measures (in *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts* **21** (5): 233-248. <http://www.sciencedirect.com/science/article/pii/0148906284926809>.
- Jaya, Balila Nagamani, Christoph Kirchlechner, Gerhard Dehm. 2015. Can microscale fracture tests provide reliable fracture toughness values? A case study in silicon (in *Journal of Materials Research* **30** (5): 686-698. <https://www.cambridge.org/core/article/can-microscale-fracture-tests-provide-reliable-fracture-toughness-values-a-case-study-in-silicon/4E645CB850A87C5DF0666BBCD93AB499>.
- Scipy: Open Source Scientific Tools for Python, 2001, "<http://www.scipy.org/>" [Online; accessed 2016-10-24].
- Koncagül, Engin C., Paul M. Santi. 1999. Predicting the unconfined compressive strength of the Breathitt shale using slake durability, Shore hardness and rock structural properties (in *International Journal of Rock Mechanics and Mining Sciences* **36** (2): 139-153. <http://www.sciencedirect.com/science/article/pii/S0148906298001740>.

- Kozicki, J, FV Donzé. 2009. Yade-open dem: an open-source software using a discrete element method to simulate granular material (in *Engineering Computations* **26** (7): 786-805.
- Kozicki, J, J Teichman. 2007. Effect of aggregate structure on fracture process in concrete using 2D lattice model (in *Archives of Mechanics* **59** (4-5): 365-384.
- Kumar, Vikas, Mark Erman Curtis, Nabanita Gupta et al. 2012. Estimation of Elastic Properties of Organic Matter in Woodford Shale Through Nanoindentation Measurements. Proc., SPE Canadian Unconventional Resources Conference, Calgary, Alberta, Canada.
- Kumar, Vikas, Carl H. Sondergeld, Chandra Shekhar Rai. 2012. Nano to Macro Mechanical Characterization of Shale. Proc., SPE Annual Technical Conference and Exhibition 2012, San Antonio, TX.
- Lempp, Ch, O Nataf, U Bayer et al. The effect of temperature on rock mechanical properties and fracture mechanisms in source rocks-Experimental results. Society of Petroleum Engineers.
- Mahabadi, O. K., N. X. Randall, Z. Zong et al. 2012. A novel approach for micro-scale characterization and modeling of geomaterials incorporating actual material heterogeneity (in *Geophysical Research Letters* **39** (1).
<http://dx.doi.org/10.1029/2011GL050411>.
- Mason, John, Joseph Carloni, Alan Zehnder et al. Dependence of Micro-Mechanical Properties on Lithofacies: Indentation Experiments on Marcellus Shale. *Denver, Colorado, USA: Unconventional Resources Technology Conference (URTEC)*.
- Mukhopadhyay, A., B. Basu. 2007. Consolidation–microstructure–property relationships in bulk nanoceramics and ceramic nanocomposites: a review (in *International Materials Reviews* **52** (5): 257-288.
<http://dx.doi.org/10.1179/174328007X160281>.
- Nasseri, M. H. B., B. Mohanty. 2008. Fracture toughness anisotropy in granitic rocks (in *International Journal of Rock Mechanics and Mining Sciences* **45** (2): 167-193.
<http://www.sciencedirect.com/science/article/pii/S1365160907000597>.
- Santi, Paul M., Jason E. Holschen, Richard W. Stephenson. 2000. Improving elastic modulus measurements for rock based on geology (in *Environmental & Engineering Geoscience* **6** (4): 333-346.
- Scholtès, Luc, Frédéric-Victor Donzé. 2012. Modelling progressive failure in fractured rock masses using a 3D discrete element method (in *International Journal of*

- Rock Mechanics and Mining Sciences* **52**: 18-30.
<http://www.sciencedirect.com/science/article/pii/S1365160912000391>.
- Scholtès, Luc, Frédéric-Victor Donzé. 2013. A DEM model for soft and hard rocks: Role of grain interlocking on strength (in *Journal of the Mechanics and Physics of Solids* **61** (2): 352-369.
<http://www.sciencedirect.com/science/article/pii/S0022509612002268>.
- Shitrit, Omri, Yossef H. Hatzor, Shimon Feinstein et al. 2016. Effect of kerogen on rock physics of immature organic-rich chalks (in *Marine and Petroleum Geology* **73**: 392-404. <http://www.sciencedirect.com/science/article/pii/S0264817216300800>.
- Sone, Hiroki, Mark D Zoback. 2013. Mechanical properties of shale-gas reservoir rocks—Part 2: Ductile creep, brittle strength, and their relation to the elastic modulus (in *Geophysics* **78** (5): D393-D402.
- Stránský, Jan, Milan Jirásek, Václav Šmilauer. Macroscopic Elastic Properties of Particle Models. In *Proceedings of the International Conference on Modelling and Simulation 2010, Proceedings of the International Conference on Modelling and Simulation 2010, Prague, Czech Republic*.
- Vásárhelyi, B. 2005. Statistical Analysis of the Influence of Water Content on the Strength of the Miocene Limestone (in *Rock Mechanics and Rock Engineering* **38** (1): 69-76. <http://dx.doi.org/10.1007/s00603-004-0034-3>.
- Voltolini, Marco, Jonathan Ajo-Franklin. 2016 -- *Personal Communication*. Digital EDS Images of Niobrara Shale, 8/17/2016.
- Wang, L. L., D. S. Yang, R. W. Yang et al. 2016. Investigating the mechanical behavior of shale: A micro-scale approach (in *Journal of Natural Gas Science and Engineering*.
<http://www.sciencedirect.com/science/article/pii/S1875510016301585>.
- Wang, XiaoLiang, JiaChun Li. 2014. Simulation of triaxial response of granular materials by modified DEM (in *Science China Physics, Mechanics & Astronomy* **57** (12): 2297-2308. <http://dx.doi.org/10.1007/s11433-014-5605-z>.
- Wang, Yuannian, Fulvio Tonon. 2010. Calibration of a discrete element model for intact rock up to its peak strength (in *International Journal for Numerical and Analytical Methods in Geomechanics* **34** (5): 447-469.
<http://dx.doi.org/10.1002/nag.811>.
- Wang, Yucang, Fernando Alonso-Marroquin. Calibration of DEM simulation: Unconfined Compressive Test and Brazilian Tensile Test. Vol. 1145, 397-400: AIP Publishing.

- Yale, D. P., W. H. Jamieson, Jr. 1994. Static and Dynamic Mechanical Properties of Carbonates. Proc. *in* P.P. Nelson and S.E. Lauboch, eds. Rock mechanics models and measurements challenges from industry. Proceedings of the 1st North American Rock Mecahnics Symposium. 463-471. Balkema.
- Zhou, Xuejun, Zhengwen Zeng, Hong Liu. Laboratory Testing on Pierre Shale for CO₂ Sequestration Under Clayey Caprock. In 44th U.S. Rock Mechancis Symposium and 4th U.S.-Canada rock mechanics Symposium. Salt Lake City, UT, 27-30 June.

APPENDIX

INSTRUCTIONS AND CODE TO RUN SIMULATION

To perform the simulations described in this thesis, the Open Source DEM framework Yade is required. In addition, we also need the open-source Scipy package and OpenCV package. This script was run on a 2013 Macbook Pro using the Ubuntu 16.04 operation system.

By default, this python script assumes the existence of SEM/EDS image files in the same directory as this script. For example, if we wished to generate a structure from the images for the images that had the prefix ‘NETL_GS_NIOB_05_’, each 272 x 187 pixels. The code assumes the existence of the following files:

‘NETL_GS_NIOB_05_SEM.png’

‘NETL_GS_NIOB_05_S.png’

‘NETL_GS_NIOB_05_O.png’

NETL_GS_NIOB_05_Ca.png’

‘NETL_GS_NIOB_05_C.png’

‘NETL_GS_NIOB_05_SEM.png’.

To actually run the code, assuming it is named “compression_script.py”, the user would enter “yadedaily compression_script.py” in their command line.

Compression_script.py:

```

from yade import pack
from yade import geom
from yade import utils
from yade import export
from yade import ymport
from yade import plot
import time
import sys
import os
import matplotlib.pyplot as plt
import numpy as np
from scipy import ndimage
import cv2
from skimage import img_as_ubyte
import csv

utils.readParamsFromTable(
    current_test_val = 0,
    oneColor = False,
    batch_single_filename_prefix = 'Example_Filename',
    temp_num = 4,
    import_name = 'NoTableDefault.notSpheres',
    max_iter = 500, #Some Maximum Iteration Value
    report_interval = 25,
    strain_rate = -5e2, #negative means compression
    stress_rate = 1e3,
    strain_limit = 1e-2,
    stress_limit = 1e4,
    number_elements = 17500, #12000,
    interactionRadius = 1.5,
    youngs_mod_clay = 1.4092e10,
    poisson_ratio_clay = 0.144,
    density_clay = 2600,
    tensile_strength_clay = 2.60160e05,
    #http://www.sciencedirect.com/science/article/pii/S0013795207001330
    youngs_mod_quartz = 2.60160E+11,
    poisson_ratio_quartz = 0.11,
    density_quartz = 2650, #quartz
    tensile_strength_quartz = 1.66123E+08,
    youngs_mod_carbonate = 2.27640E+11,
    poisson_ratio_carbonate = 0.17,
    density_carbonate = 2160, #limestone
    tensile_strength_carbonate = 2.71e6,
    damping = 0.2,
    ITZ_override_mat = 'clay',
    ITZ_length = 0e-5,
    youngs_mod_ITZ = 9e8,
    poisson_ratio_ITZ = 0.144,
    density_ITZ = 2600,
    tensile_strength_ITZ = 1.92e4,
    cohesion_strength_ITZ = 1.92e4,
    modify_ITZ_bond_strength = False,
    weaken_clay_quartz_strength = False,

```

```

    ITZ_bond_weaken_ratio = 1.0
)
from yade.params import table

#Now create all the various output names
print("prefix is: " + table.batch_single_filename_prefix)
triax_cracks_filename = table.batch_single_filename_prefix + "_fractures_" + str(table.temp_num)
triax_stress_strain_filename = table.batch_single_filename_prefix + "_stress_strain_" +
str(table.temp_num) + ".csv"
cracks_filename_info = table.batch_single_filename_prefix + "_overlaid_fractures_" +
str(table.temp_num)+".png"
summary_filename = table.batch_single_filename_prefix + "_summary.csv" #currently only for uniaxial

#other random parameters
smoothContact=True
jointFrict=radians(20)
jointDil=radians(0)
setSpeeds=True

class test_identifiers:
    def __init__(self, prefix, test_num, test_scale, pixel_val):
        self.prefix = prefix
        self.test_num = test_num
        self.test_scale = test_scale
        self.pixel_len = pixel_val

EDS_test_vals = [
    test_identifiers('NETL_GS_NIOB_05_', '05', 25e-6/83, 1e-6),
    test_identifiers('NETL_GS_NIOB_07_', '07', 50e-6/61, 2.7e-6),
    test_identifiers('NETL_GS_NIOB_09_', '09', 500e-6/94, 1.766e-5),
    test_identifiers('NETL_GS_NIOB_11_', '11', 100e-6/52, 6.3846e-6)
]

##### DECIDE WHICH TEST TO EXAMINE HERE!!! #####
current_test = EDS_test_vals[table.current_test_val]
EDS_scale = current_test.test_scale
prefix=current_test.prefix
suffix = '.png'

print("Size scale is: " + str(EDS_scale) + " m per pixel")

start = time.time()

#this function only for growth function
def stop_at_porosity(porosity_limit, porosity_list, delta_porosity_limit = 1e-3):
    porosity_list.append(utils.porosity())
    if(porosity_limit > porosity_list[-1]):

```

```

        print('porosity limit reached, stopping now!')
        O.pause()
    elif( (porosity_list[-2] - porosity_list[-1]) < delta_porosity_limit):
        print('change in porosity too small! Stopping now!')
        O.pause()
    else:
        print("porosity is: " + str(porosity_list[-1]))

porosity_list = [1.0]

#this function just makes a sphere mesh. It was adapted from the Yade TriaxStressController example
package
def Make_sphere_mesh_file_growth(filename, xSize, ySize, num_spheres, rRelFuzz_value, porosity_limit
= 0.46):
    young=9e5 # contact stiffness
    mn,mx=Vector3(0,0,0),Vector3(xSize,ySize,0) # corners of the initial packing

O.materials.append(FrictMat(young=young,poisson=0.5,frictionAngle=radians(30),density=2600,label='s
pheres'))
    O.materials.append(FrictMat(young=young,poisson=0.5,frictionAngle=0,density=0,label='walls'))
    sp=pack.SpherePack()
    sphererad = 0
    sp.makeCloud(mn,mx,-1,0,0,num_spheres,False, 0.95) #"seed" make the "random" generation always
the same
    for center, rad in sp:
        sphererad = rad
        break
    wall_mn, wall_mx = Vector3(-sphererad,-sphererad,-
1000*sphererad),Vector3(xSize+sphererad,ySize+sphererad,1000*sphererad)
    walls=aabbWalls([ wall_mn,wall_mx],thickness=0,material='walls')
    wallIds=O.bodies.append(walls)
    O.bodies.append([sphere(center,rad,material='spheres') for center,rad in sp])
    triax=TriaxialStressController(
    maxMultiplier=1.+2e4/young, # spheres growing factor (fast growth)
    finalMaxMultiplier=1.+2e3/young, # spheres growing factor (slow growth)
    thickness = 0,
    stressMask = 7,
    internalCompaction=True, # If true the confining pressure is generated by growing particles
    )
    newton=NewtonIntegrator(damping=0.2)
    O.engines=[
        ForceResetter(),
        InsertionSortCollider([Bo1_Sphere_Aabb(),Bo1_Box_Aabb()]),
        InteractionLoop(
            [Ig2_Sphere_Sphere_ScGeom(),Ig2_Box_Sphere_ScGeom()],
            [Ip2_FrictMat_FrictMat_FrictPhys()],
            [Law2_ScGeom_FrictPhys_CundallStrack()]
        ),
        ## We will use the global stiffness of each body to determine an optimal timestep (see https://yade-
dem.org/w/images/1/1b/Chareyre&Villard2005_licensed.pdf)
        GlobalStiffnessTimeStepper(active=1,timeStepUpdateInterval=100,timestepSafetyCoefficient=0.8),
        triax,

```

```

        PyRunner(iterPeriod=500,initRun=False,command='stop_at_porosity(' + str(porosity_limit) + ',
porosity_list)'),
        newton
    ]
    triax.goal1=triax.goal2=triax.goal3=-5
    O.run(15001)
    O.wait()
    radius = O.bodies[6].shape.radius;
    print("Growth packing time = " + str(time.time()-start) + ", total porosity: " + str(utils.porosity()) + ",
radius: " + str(radius) )
    export.text(filename+'.spheres')

rRelFuzz_value = 0;
EDS_xCorner = 5;
EDS_yCorner = 33;
EDS_xSize = 272;
EDS_ySize = 187;

SEM_xCorner = 0;
SEM_yCorner = 0;
SEM_xSize = 272;
SEM_ySize = 187;

Make_sphere_mesh_file_growth("packed_2D_mesh_"+prefix+str(start), EDS_xSize, EDS_ySize,
table.number_elements, rRelFuzz_value)

O.reset() #reset everything, just in case

O_thresh = 30;
C_thresh = 30;
Si_thresh = 30;
Ca_thresh = 30;
S_thresh = 30;
Fe_thresh = 30;
SEM_thresh = 16;

face_O = ndimage.imread(prefix+'O'+suffix,True)
face_C = ndimage.imread(prefix+'C'+suffix,True)
face_Si = ndimage.imread(prefix+'Si'+suffix,True)
face_Ca = ndimage.imread(prefix+'Ca'+suffix,True)
face_S = ndimage.imread(prefix+'S'+suffix,True)
#face_Fe = ndimage.imread(prefix+'Fe'+suffix,True)
face_SEM = ndimage.imread(prefix+'SEM'+suffix,True)

face_O_cropped = face_O[EDS_yCorner:(EDS_yCorner+EDS_ySize),
EDS_xCorner:(EDS_xCorner+EDS_xSize)];
face_C_cropped = face_C[EDS_yCorner:(EDS_yCorner+EDS_ySize),
EDS_xCorner:(EDS_xCorner+EDS_xSize)];
face_Si_cropped = face_Si[EDS_yCorner:(EDS_yCorner+EDS_ySize),
EDS_xCorner:(EDS_xCorner+EDS_xSize)];
face_Ca_cropped = face_Ca[EDS_yCorner:(EDS_yCorner+EDS_ySize),
EDS_xCorner:(EDS_xCorner+EDS_xSize)];

```



```

face_S_cropped = face_S[EDS_yCorner:(EDS_yCorner+EDS_ySize),
EDS_xCorner:(EDS_xCorner+EDS_xSize)];
#face_Fe_cropped = face_Fe[EDS_yCorner:(EDS_yCorner+EDS_ySize),
EDS_xCorner:(EDS_xCorner+EDS_xSize)];
face_SEM_cropped = face_SEM[SEM_yCorner:(SEM_yCorner+SEM_ySize),
SEM_xCorner:(SEM_xCorner+SEM_xSize)];

binary_O = (face_O_cropped > O_thresh)
open_img_O = ndimage.binary_opening(binary_O)
close_img_O = ndimage.binary_closing(open_img_O)

binary_C = (face_C_cropped > C_thresh)
open_img_C = ndimage.binary_opening(binary_C)
close_img_C = ndimage.binary_closing(open_img_C)

binary_Si = (face_Si_cropped > Si_thresh)
open_img_Si = ndimage.binary_opening(binary_Si)
close_img_Si = ndimage.binary_closing(open_img_Si)

binary_Ca = (face_Ca_cropped > Ca_thresh)
open_img_Ca = ndimage.binary_opening(binary_Ca)
close_img_Ca = ndimage.binary_closing(open_img_Ca)

binary_S = (face_S_cropped > S_thresh)
open_img_S = ndimage.binary_opening(binary_S)
close_img_S = ndimage.binary_closing(open_img_S)

binary_SEM = (face_SEM_cropped < SEM_thresh)
#open_img_SEM = ndimage.binary_opening(binary_SEM)
#close_img_SEM = ndimage.binary_closing(open_img_SEM)

#binary_Fe = (face_Fe_cropped > Fe_thresh)
#open_img_Fe = ndimage.binary_opening(binary_Fe)
#close_img_Fe = ndimage.binary_closing(open_img_Fe)

#create a material array:
#Si -> Quartz
#S -> Pyrite, but define as Quartz
#Ca & O together -> Carbonate
#C -> kerogen
# the rest -> clay

#now that the four channels are input:

class Material_Voxel_Map:
    def __init__(self, array_size, channels_array_dict, channel_names, matNum, mat_dictionary =
{'empty':0, 'kerogen':10, 'clay':20, 'carbonate':30, 'quartz':40}):
        self.array_size = array_size
        self.n_channels = len(channels_array_dict)
        self.num_matIDs = matNum
        self.channels_array_dict = channels_array_dict

```

```

self.channel_names = channel_names
self.mat_dict = mat_dictionary
self.gotten_final_map = False
self.final_map_2d = numpy.zeros(array_size, np.uint8)
self.final_map_with_interfaces = numpy.zeros(array_size, np.uint8)
#here, I make a function method that will assign the value of the voxel
#This assumes that all of the channels have been aligned and are of the same size
#This also assumes that x_val and y_val are already in the array
def set_voxel_matID(self, y_val,x_val):
    if (x_val == 0):
        self.final_map_2d[y_val][x_val] = self.mat_dict['quartz']
    elif self.channels_array_dict['SEM'][y_val][x_val]:
        self.final_map_2d[y_val][x_val] = self.mat_dict['kerogen'] #actually empty space, use for kerogen
as void space
    elif (self.channels_array_dict['S'][y_val][x_val]):
        #self.final_map_2d[y_val][x_val] = self.mat_dict['pyrite'] #should be checking for iron or sulfur
content
        self.final_map_2d[y_val][x_val] = self.mat_dict['quartz'] #for calculation simplicity, call it quartz
    elif self.channels_array_dict['O'][y_val][x_val]:
        if self.channels_array_dict['Si'][y_val][x_val]:
            self.final_map_2d[y_val][x_val] = self.mat_dict['quartz']
        elif (self.channels_array_dict['Ca'][y_val][x_val] and self.channels_array_dict['O'][y_val][x_val]):
            self.final_map_2d[y_val][x_val] = self.mat_dict['carbonate']
        else: #by default, if there is oxygen and nothing else, it's probably an oxide clay
            self.final_map_2d[y_val][x_val] = self.mat_dict['clay']
    elif (self.channels_array_dict['C'][y_val][x_val]):
        self.final_map_2d[y_val][x_val] = self.mat_dict['kerogen']
    else: #it's either clay or empty, depending of SEM
        self.final_map_2d[y_val][x_val] = self.mat_dict['clay'] #clay is just the filler material for
everything else

def apply_rules(self):
    for y_pos in range(self.array_size[0]):
        for x_pos in range(self.array_size[1]):
            self.set_voxel_matID(y_pos,x_pos)
    self.gotten_final_map = True
    self.final_map_with_interfaces = self.final_map_2d.copy()
    return self.final_map_2d

def get_voxel_matID(self,y_val, x_val):
    #print("x-val is: " + str(x_val) + ", y-val is: " + str(y_val) )
    return self.final_map_with_interfaces[y_val][x_val]

def get_voxel_matID_orig(self,y_val, x_val):
    #print("x-val is: " + str(x_val) + ", y-val is: " + str(y_val) )
    return self.final_map_2d[y_val][x_val]

def isolate_material_image(self, material_name):
    if (self.gotten_final_map == False):
        print("need to get the final map first. Run the 'apply_rules' method first.")
    else:
        temp_array = numpy.zeros([self.array_size[0], self.array_size[1]], np.uint8)
        for y_pos in range(self.array_size[0]):

```

```

        for x_pos in range(self.array_size[1]):
            if (self.final_map_2d[y_pos][x_pos] == self.mat_dict[material_name]):
                temp_array[y_pos][x_pos] = 255
        ret, temp_array = cv2.threshold(temp_array, 1, 255, 0)
        return temp_array
    def find_combined_edges(self, mat_name_1 = 'quartz', mat_name_2 = 'clay', thickness=1, target = 2):
        contours_1 = self.isolate_material_image(mat_name_1)
        contours_2 = self.isolate_material_image(mat_name_2)
        contours1, hierarchy =
cv2.findContours(contours_1, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
        contours2, hierarchy =
cv2.findContours(contours_2, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
        cv2.drawContours(contours_1, contours1, -1, (100,100,0), thickness)
        cv2.drawContours(contours_2, contours2, -1, (100,100,0), thickness)
        temp_array = numpy.zeros([self.array_size[0], self.array_size[1]], np.uint8)
        for y_pos in range(self.array_size[0]):
            for x_pos in range(self.array_size[1]):
                #print("edge_array1[y_pos][x_pos] is: " + str(contours_1[y_pos][x_pos]) + ",
edge_array2[y_pos][x_pos] is: " + str(contours_2[y_pos][x_pos]) )
                if ( (contours_1[y_pos][x_pos] > 0) and (contours_2[y_pos][x_pos] > 0) ):
                    if (target == 1):
                        if (self.final_map_2d[y_pos][x_pos] == self.mat_dict[mat_name_1]):
                            temp_array[y_pos][x_pos] = 255
                    elif (target == 2):
                        if (self.final_map_2d[y_pos][x_pos] == self.mat_dict[mat_name_2]):
                            temp_array[y_pos][x_pos] = 255
                    else:
                        temp_array[y_pos][x_pos] = 255
        return temp_array

    def add_interface_layer(self, mat1='quartz', mat2='clay', thickness=2, replacement_value1 = 39,
replacement_value2 = 15):
        self.final_map_with_interfaces = self.final_map_with_interfaces.copy()
        temp_array1 = self.find_combined_edges(mat1, mat2, thickness, 1)
        temp_array2 = self.find_combined_edges(mat1, mat2, thickness, 2)
        for y_pos in range(self.array_size[0]):
            for x_pos in range(self.array_size[1]):
                if (temp_array1[y_pos][x_pos] > 0): self.final_map_with_interfaces[y_pos][x_pos] =
replacement_value1
                if (temp_array2[y_pos][x_pos] > 0): self.final_map_with_interfaces[y_pos][x_pos] =
replacement_value2
        return self.final_map_with_interfaces

mat_map = Material_Voxel_Map( [EDS_ySize, EDS_xSize],
    {'Si':close_img_Si, 'Ca':close_img_Ca, 'O':close_img_O, 'C': close_img_C, 'S':close_img_S,
'SEM':binary_SEM},
    ['Si', 'Ca', 'O', 'C', 'S'],
    5 );

final_map_no_interfaces = mat_map.apply_rules()

#add the 1-pixel interface layer for both carbonate and quartz.

```

```

integrated_edges = mat_map.add_interface_layer('quartz', 'clay', ITZ_length_in_pixels, 39, 15)
integrated_edges2 = mat_map.add_interface_layer('carbonate', 'clay', ITZ_length_in_pixels, 29, 15)

#def mat(): return
JCFpmMat(type=1,young=1e8,poisson=0.3,frictionAngle=radians(30),density=3000,tensileStrength=1e6,
cohesion=1e6,jointNormalStiffness=1e7,jointShearStiffness=1e7,jointCohesion=1e6,jointFrictionAngle=r
adians(20),jointDilationAngle=0.0)

def mat_clay(): return JCFpmMat(type=1,
    young=table.youngs_mod_clay,
    poisson=table.poisson_ratio_clay,
    frictionAngle=radians(30),
    density=table.density_clay,
    tensileStrength=table.tensile_strength_clay,
    cohesion=0.5*table.tensile_strength_clay,
    jointNormalStiffness=1e7,
    jointShearStiffness=1e7,
    jointCohesion=1e6,
    jointFrictionAngle=radians(20),
    jointDilationAngle=0.0,
    label = 'clay')

def mat_carbonate(): return JCFpmMat(type=1,
    young=table.youngs_mod_carbonate,
    poisson=table.poisson_ratio_carbonate,
    frictionAngle=radians(30),
    density=table.density_carbonate,
    tensileStrength=table.tensile_strength_carbonate,
    cohesion=0.5*table.tensile_strength_carbonate,
    jointNormalStiffness=1e7,
    jointShearStiffness=1e7,
    jointCohesion=1e6,
    jointFrictionAngle=radians(20),
    jointDilationAngle=0.0,
    label = 'carbonate')

def mat_ITZ_carbonate(): return JCFpmMat(type=1,
    young=table.youngs_mod_carbonate,
    poisson=table.poisson_ratio_carbonate,
    frictionAngle=radians(30),
    density=table.density_carbonate,
    tensileStrength=table.tensile_strength_carbonate,
    cohesion=0.5*table.tensile_strength_carbonate,
    jointNormalStiffness=1e7,
    jointShearStiffness=1e7,
    jointCohesion=1e6,
    jointFrictionAngle=radians(20),
    jointDilationAngle=0.0,
    label = 'ITZ_carbonate')

def mat_quartz(): return JCFpmMat(type=1,
    young=table.youngs_mod_quartz,

```

```

    poisson=table.poisson_ratio_quartz,
    frictionAngle=radians(30),
    density=table.density_quartz,
    tensileStrength=table.tensile_strength_quartz,
    cohesion=0.5*table.tensile_strength_quartz,
    jointNormalStiffness=1e7,
    jointShearStiffness=1e7,
    jointCohesion=1e6,
    jointFrictionAngle=radians(20),
    jointDilationAngle=0.0,
    label = 'quartz') #UTS from http://www.mt-berlin.com/frames_cryst/descriptions/quartz%20.htm

def mat_ITZ_quartz(): return JCFpmMat(type=1,
    young=table.youngs_mod_quartz,
    poisson=table.poisson_ratio_quartz,
    frictionAngle=radians(30),
    density=table.density_quartz,
    tensileStrength=table.tensile_strength_quartz,
    cohesion=0.5*table.tensile_strength_quartz,
    jointNormalStiffness=1e7,
    jointShearStiffness=1e7,
    jointCohesion=1e6,
    jointFrictionAngle=radians(20),
    jointDilationAngle=0.0,
    label = 'ITZ_quartz') #UTS from http://www.mt-berlin.com/frames_cryst/descriptions/quartz%20.htm

def mat_pyrite(): return JCFpmMat(type=1,
    young=table.youngs_mod_quartz,
    poisson=table.poisson_ratio_quartz,
    frictionAngle=radians(30),
    density=table.density_quartz,
    tensileStrength=table.tensile_strength_quartz,
    cohesion=0.5*table.tensile_strength_quartz,
    jointNormalStiffness=1e7,
    jointShearStiffness=1e7,
    jointCohesion=1e6,
    jointFrictionAngle=radians(20),
    jointDilationAngle=0.0,
    label = 'pyrite') #filler entry for pyrite

if(table.ITZ_override_mat == 'clay'):
    def mat_ITZ_clay(): return JCFpmMat(type=1,
        young=table.youngs_mod_clay,
        poisson=table.poisson_ratio_clay,
        frictionAngle=radians(30),
        density=table.density_clay,
        tensileStrength=table.tensile_strength_clay,
        cohesion=0.5*table.tensile_strength_clay,
        jointNormalStiffness=1e7,
        jointShearStiffness=1e7,
        jointCohesion=1e6,
        jointFrictionAngle=radians(20),

```

```

        jointDilationAngle=0.0,
        label = 'ITZ_clay')
else:
    def mat_ITZ_clay(): return JCFpmMat(type=1,
        young=table.youngs_mod_ITZ,
        poisson=table.poisson_ratio_clay,
        frictionAngle=radians(30),
        density=table.density_ITZ,
        tensileStrength=table.tensile_strength_ITZ,
        cohesion=0.5*table.tensile_strength_ITZ,
        jointNormalStiffness=1e7,
        jointShearStiffness=1e7,
        jointCohesion=1e6,
        jointFrictionAngle=radians(20),
        jointDilationAngle=0.0,
        label = 'ITZ_clay')

O.materials.append( (mat_clay(), mat_clay(), mat_carbonate(), mat_quartz(), mat_ITZ_clay(),
mat_ITZ_carbonate(), mat_ITZ_quartz() ) )

#need to create a set of arrays representing the coordinates of each material group
#We should have 6 materials (Empty, Kerogen, Clay, Carbonate, Quartz, and Pyrite).
#For each imported point, we will assign material id.

if (table.import_name != "NoTableDefault.notSpheres"):
    import_filename = table.import_name
else:
    import_filename = "packed_2D_mesh_"+prefix + str(start) + ".spheres"

import_spheres = ymport.text("packed_2D_mesh_"+prefix + str(start) +
".spheres",scale=EDS_scale,shift=Vector3(0,0,0),material=mat_clay)
os.remove("packed_2D_mesh_"+prefix + str(start) + ".spheres")

print("number of materials identified (not including ITZ): " + str(mat_map.num_matIDs))

colorRed=(1,0,0)# ---red color
colorGreen=(0,1,0)# ----green color
colorBlue=(0,0,1)# ---blue color
colorBlack = (0.15,0.15,0.15)# ---black color
colorYellow = (1, 1, 0)# ---yellow color
colorPurple = (.5, 0, .5)
colorOrange = (1, 153.0/255, 0)
colorOrange_ITZ = (235.0/255, 133.0/255, 0)
colorLightBlueGreen = (100.0/255, 1, 127.0/255)
colorLightBlueGreen_ITZ = (0, 1, 122.0/255)
colorDeepSkyBlue = (60.0/255, 140.0/255, 1)
colorDeepSkyBlue_ITZ = (0, 181.0/255, 245.0/255)
colorFireBrick = (165.0/255, 42.0/255, 42.0/255)

```

```

if (table.oneColor == True):
    colorMap={ 10:colorDeepSkyBlue, 15: colorDeepSkyBlue, 20:colorDeepSkyBlue, 30:
colorDeepSkyBlue, 40: colorDeepSkyBlue, 29: colorDeepSkyBlue, 39: colorDeepSkyBlue}
else:
    colorMap={ 10:colorBlack, 15: colorBlue, 20:colorDeepSkyBlue, 30: colorLightBlueGreen, 40:
colorOrange, 29: colorLightBlueGreen_ITZ, 39:colorOrange_ITZ}

#Following class contains all data about the initial material ID
#Also contains any joints/facet data
class Spheres_Mat_Structure:
    def __init__(self, mat_names_array, matID_to_Ind_Dict,import_spheres, mat_map_array, color_map,
size_scale, maxX, maxY):
        self.num_matIDs = len(mat_names_array)
        self.mat_names_array = mat_names_array
        self.import_spheres = import_spheres #the imported spheres
        self.el_matID = [[]]; #want each entry to have the sphere ID and the mat ID associated with it
        for i in range(self.num_matIDs-1): self.el_matID.append([])
        self.matID_to_Ind_Dict = matID_to_Ind_Dict #{10:0, 20:1, 30:2, 40:3, 50:4}
        self.color_map = color_map
        self.max_X = 0;
        self.max_Y = 0;
        self.max_Z = 0;
        self.min_X = 0;
        self.min_Y = 0;
        self.min_Z = 0;
        self.radius = import_spheres[0].shape.radius;
        self.total_mass = 0;
        self.num_spheres = 0;
        for sphere in self.import_spheres:
            temp_x_pos = int((sphere.state.pos[0]/size_scale)) #need to subtract by two because the importer is
imperfect
            temp_y_pos = int((sphere.state.pos[1]/size_scale)) #need to subtract by two because the importer is
imperfect
            self.num_spheres += 1
            if (self.max_X < sphere.state.pos[0]): self.max_X = sphere.state.pos[0]
            if (self.max_Y < sphere.state.pos[1]): self.max_Y = sphere.state.pos[1]
            if (self.max_Z < sphere.state.pos[2]): self.max_Z = sphere.state.pos[2]
            if (self.min_X > sphere.state.pos[0]): self.min_X = sphere.state.pos[0]
            if (self.min_Y > sphere.state.pos[1]): self.min_Y = sphere.state.pos[1]
            if (self.min_Z > sphere.state.pos[2]): self.min_Z = sphere.state.pos[2]
            sphere.mat = O.materials[matID_to_Ind_Dict[mat_map_array[temp_y_pos][temp_x_pos]]]
            sphere.shape.color = color_map[mat_map_array[temp_y_pos][temp_x_pos]]
            self.el_matID[matID_to_Ind_Dict[mat_map_array[temp_y_pos][temp_x_pos]]].append(sphere)
        print("number of spheres imported: " + str(self.num_spheres) )
        self.max_X += self.radius
        self.max_Y += self.radius*2.5
        self.max_Z += self.radius*2.5
        self.min_X -= self.radius
        self.min_Y -= self.radius*2.5
        self.min_Z -= self.radius*2.5

```

```

true_EDS_scale = 1.005*EDS_scale

sphere_mat_structure = Spheres_Mat_Structure(['kerogen', 'clay', 'carbonate', 'quartz', 'ITZ',
'ITZ_carbonate', 'ITZ_quartz'], {10:0, 20:1, 30:2, 40:3, 15:4, 29:5, 39:6}, import_spheres,
integrated_edges2, colorMap, true_EDS_scale, mat_map.array_size[1], mat_map.array_size[0]);

#for mat_ind in range(4):
#   for spheres in sphere_mat_structure.el_matID[mat_ind]:
#       continue
#       #O.bodies.append(import_spheres)

##PRE-PROCESSING STEP
## preprocessing to get dimensions of the packing
dim=utils.aabbExtrema()
xinf=dim[0][0]
xsup=dim[1][0]
X=xsup-xinf
yinf=dim[0][1]
ysup=dim[1][1]
Y=ysup-yinf
zinf=dim[0][2]
zsup=dim[1][2]
Z=zsup-zinf

##### walls generation #####
#add in wall information, from uniax.py
young = 9e12
mn = Vector3(sphere_mat_structure.min_X,sphere_mat_structure.min_Y,sphere_mat_structure.min_Z)
mx = Vector3(sphere_mat_structure.max_X,sphere_mat_structure.max_Y,sphere_mat_structure.max_Z) #
corners of the initial packing

print("mn is: " + str(mn))
print("mx is: " + str(mx))

def mat_wall(): return
JCFpmMat(type=1,young=young,poisson=0.3,frictionAngle=radians(30),density=3000,tensileStrength=young,cohesion=young,jointNormalStiffness=1e8,jointShearStiffness=1e8,jointCohesion=1e7,jointFrictionAngle=radians(20),jointDilationAngle=0.0,label='walls')
O.materials.append((mat_wall()))
#walls=aabbWalls([mn,mx],thickness=0,material=mat_wall)
#wallIds=O.bodies.append(walls)

## preprocessing to get spheres dimensions

#add all elements to simulation
for i in range(1,sphere_mat_structure.num_matIDs):
    O.bodies.append(sphere_mat_structure.el_matID[i])

for i in range(0,sphere_mat_structure.num_matIDs):

```



```

    print("element type: " + sphere_mat_structure.mat_names_array[i] + ", count is: " +
    str(len(sphere_mat_structure.el_matID[i])))

#Get a count of all the different element types
kerogen_count = len(sphere_mat_structure.el_matID[0])
clay_count = len(sphere_mat_structure.el_matID[1])
carbonate_count = len(sphere_mat_structure.el_matID[2])
quartz_count = len(sphere_mat_structure.el_matID[3])
ITZ_clay_count = len(sphere_mat_structure.el_matID[4])
ITZ_carbonate_count = len(sphere_mat_structure.el_matID[5])
ITZ_quartz_count = len(sphere_mat_structure.el_matID[6])
total_element =
clay_count+carbonate_count+quartz_count+ITZ_clay_count+ITZ_carbonate_count+ITZ_quartz_count

def AddStressStrain(uniax, trial_data):
    if( (uniax.avgStress < trial_data.stress_limit) and (uniax.strain > trial_data.strain_limit) ):
        trial_data.append_stress_and_strain(uniax.avgStress, uniax.strain)
        O.pause()
    else:
        trial_data.append_stress_and_strain(uniax.avgStress, uniax.strain)
        #print(str(round(time.time() - start,1))) + "," + str(O.iter) + ", uniax.avgStress: " +
        str(round(uniax.avgStress,1))
        #print(str(round(time.time() - start,1))) + "," + str(O.iter) + ", trial_data.curr_stress_strain_point[0]" +
        str(round(trial_data.curr_stress_strain_point[0],1))
        #print(str(round(time.time() - start,1))) + "," + str(O.iter) + ", " + str(uniax.strain) + ", " +
        str(round(uniax.avgStress,1))

def increase_strain_rate(multiplier):
    uniax.strainRate = multiplier*uniax.strainRate

def print_and_stop():
    print("triax.meanstress = " + str(triax.meanStress) )
    if (triax.meanStress == 0.0):
        O.pause()

stressList = [];
strainList = [];

def add_line_to_file(stored_name, stressList):
    column_headers = "Test Elapsed Time;Iteration;Strain;Average Stress"
    delimiter = ";"
    if (table.strain_rate < 0):
        data_seq = [round(time.time() - start,1), O.iter, -uniax.strain, -round(uniax.avgStress,1)]
    else:
        data_seq = [round(time.time() - start,1), O.iter, uniax.strain, round(uniax.avgStress,1)]
    for i in range(len(data_seq)): data_seq[i] = str(data_seq[i])
    append_line = delimiter.join(data_seq)
    ##### actually print data output
    if (os.path.exists(stored_name)):
        myfile = open(stored_name, "a")

```

```

        #print("already exists, appending")
        myfile.write("\n")
        myfile.write(append_line)
        myfile.close()
        stressList.append(-uniax.avgStress)
        strainList.append(-uniax.strain)
    else:
        myfile = open(stored_name, "a")
        print("doesn't exist, creating a new file with filename = " + stored_name)
        myfile.write("sep=;")
        myfile.write("\n")
        myfile.write(column_headers)
        myfile.write("\n")
        myfile.write(append_line)
        myfile.close()
        stressList.append(-uniax.avgStress)
        strainList.append(-uniax.strain)

PRCom_add_csv = 'add_line_to_file(table.batch_single_filename_prefix + "_" + str(table.temp_num) +
".csv",stressList)'
PRCom_stop_at_strain_lim = 'stop_at_strain_limit_uniax(table.strain_limit, strainList)'

num_data_pts = int(table.max_iter/table.report_interval)+1

curr_strainLimit = table.strain_limit
curr_stressLimit = table.stress_limit
#trial01 = TrialData(1, strainRateTension, num_data_pts, curr_strainLimit, curr_stressLimit)

#for uniax
bb=uniaxialTestFeatures()
negIds,posIds,axis,crossSectionArea=bb['negIds'],bb['posIds'],bb['axis'],bb['area']

strainRateTension=table.strain_rate

#this function stops at strain function
def stop_at_strain_limit_uniax(strain_limit, strain_list):
    if(strain_limit < strain_list[-1]):
        print('strain limit of' + str(strain_limit) + ' reached, stopping now!')
        O.pause()

def stop_at_stress_limit_uniax(stress_limit, stress_list):
    if(stress_limit < stress_list[-1]):
        print('stress limit of' + str(stress_limit) + ' reached, stopping now!')
        O.pause()

uniax = UniaxialStrainer(
    strainRate=strainRateTension,
    axis=axis,
    asymmetry=0,

```

```

posIds=posIds,
negIds=negIds,
crossSectionArea=crossSectionArea,
blockDisplacements=False,
blockRotations=False,
setSpeeds=setSpeeds,
label='strainer')

O.engines=[

    ForceResetter(),

    InsertionSortCollider([Bo1_Sphere_Aabb(aabbEnlargeFactor=table.interactionRadius,label='is2aabb'),Bo
1_Box_Aabb()]),
    InteractionLoop(
        [Ig2_Sphere_Sphere_ScGeom(interactionDetectionFactor=table.interactionRadius,label='ss2d3dg'),
Ig2_Box_Sphere_ScGeom()],
        [Ip2_JCFpmMat_JCFpmMat_JCFpmPhys(cohesiveTresholdIteration=1,label='interactionPhys')],

        [Law2_ScGeom_JCFpmPhys_JointedCohesiveFrictionalPM(smoothJoint=smoothContact,label='interacti
onLaw',recordCracks=True,Key=triax_cracks_filename)]
    ),
    GlobalStiffnessTimeStepper(timestepSafetyCoefficient=0.8),
    uniax, #this is unrealistic straining
    PyRunner(iterPeriod=table.report_interval,initRun=False,command=PRCom_add_csv),
    PyRunner(iterPeriod=table.report_interval,initRun=False,command=PRCom_stop_at_strain_lim),
    NewtonIntegrator(damping=0.2,gravity=(0.,0.0.))
]
#### time step definition (low here to create cohesive links without big changes in the assembly)
O.dt=0.25*utils.PWaveTimeStep()

#### set cohesive links with interaction radius>=1
O.step();

#### initializes now the interaction detection factor to strictly 1
ss2d3dg.interactionDetectionFactor=1.0
is2aabb.aabbEnlargeFactor=1.0

weakenedCount = 0;
unWeakenedCount = 0;

num_wall_interactions = 0;

#count each interaction to determine if wall-particle interactions exist
check_wall_interactions = True
double_wall_count = 0
if check_wall_interactions:
    for i in O.interactions:
        last_i = i
        mat1 = O.bodies[i.id1].mat
        mat2 = O.bodies[i.id2].mat
        if (i.phys.isCohesive == True):

```

```

        if((mat1.label == 'walls') and (mat2.label == 'walls') ):
            double_wall_count +=1
        elif((mat1.label == 'walls') or (mat2.label == 'walls') ):
            num_wall_interactions += 1

print("number of wall interactions: " + str(num_wall_interactions) )
print("number of double wall interactions: " + str(double_wall_count) )
print("weakened: " + str(weakenedCount))
print("Remained the same: " + str(unWeakenedCount) )
print("elapsed time after checking interactions: " + str(time.time() - start))

#iterate over every wall-to-material to increase the strength of inter-material bonds
modify_wall_interfaces = False
wall_strengthened_num = 0
if modify_wall_interfaces:
    for i in O.interactions:
        last_i = i
        mat1 = O.bodies[i.id1].mat
        mat2 = O.bodies[i.id2].mat
        if(mat1.label == 'walls'):
            i.phys.FnMax = 1000*i.phys.FnMax
            i.phys.kn = i.phys.kn * 1000 * mat1.young
            wall_strengthened_num += 1
        elif(mat2.label == 'walls'):
            i.phys.FnMax = 1000*i.phys.FnMax
            i.phys.kn = i.phys.kn * 1000 * mat2.young
            wall_strengthened_num += 1

print("wall strengthened number: " + str(wall_strengthened_num))

#iterate over every ITZ-to-material to reduce the strength of inter-material bonds
if table.modify_ITZ_bond_strength:
    default_fnmax = 0;
    for i in O.interactions:
        last_i = i
        mat1 = O.bodies[i.id1].mat
        mat2 = O.bodies[i.id2].mat
        if((mat1.label == 'ITZ') and (mat2.label == 'ITZ')):
            default_fnmax = i.phys.FnMax
            break
    for i in O.interactions:
        last_i = i
        mat1 = O.bodies[i.id1].mat
        mat2 = O.bodies[i.id2].mat
        if((mat1.label == 'quartz') and (mat2.label == 'ITZ')):
            i.phys.FnMax = table.ITZ_bond_weaken_ratio * default_fnmax
        elif((mat1.label == 'ITZ') and (mat2.label == 'quartz')):
            i.phys.FnMax = table.ITZ_bond_weaken_ratio * default_fnmax

modify_strength_ratio = 1
modify_ITZ_quartz_interfaces2 = False

```

```

if modify_ITZ_quartz_interfaces2:
    for i in O.interactions:
        last_i = i
        mat1 = O.bodies[i.id1].mat
        mat2 = O.bodies[i.id2].mat
        if((mat1.label == 'quartz') and (mat2.label == 'quartz')):
            i.phys.FnMax = modify_strength_ratio * i.phys.FnMax
            print("quartz - quartz is: " + str(i.phys.FnMax))
            print("\n")
        if((mat1.label == 'ITZ') and (mat2.label == 'ITZ')):
            i.phys.FnMax = modify_strength_ratio * i.phys.FnMax
            print("ITZ - ITZ is: " + str(i.phys.FnMax))
            print("\n")
        if((mat1.label == 'quartz') and (mat2.label == 'ITZ')):
            i.phys.FnMax = modify_strength_ratio * i.phys.FnMax
            print("quartz - ITZ is: " + str(i.phys.FnMax))
            print("\n")
        elif((mat1.label == 'ITZ') and (mat2.label == 'quartz')):
            i.phys.FnMax = modify_strength_ratio * i.phys.FnMax
            #i.phys.kn = i.phys.kn * 1000 * mat2.young
            print("ITZ - quartz is: " + str(i.phys.FnMax))
            print("\n")

#iterate over every interaction to reduce the strength of inter-material bonds
modify_interfaces = False
if modify_interfaces:
    for i in O.interactions:
        last_i = i
        mat1 = O.bodies[i.id1].mat
        mat2 = O.bodies[i.id2].mat
        if(mat1 != mat2):
            #i.phys.kn = i.phys.kn
            i.phys.FnMax = i.phys.FnMax
            weakenedCount += 1
        else:
            unWeakenedCount += 1
            #E1,E2 = mat1.young, mat2.young
            #i.phys.kn = i.phys.crossSection * 0.5*min(E1, E2) / i.phys.refLength

#iterate over every interaction to reduce the strength of inter-material bonds
make_borders_strong = True
if modify_interfaces:
    buffer_range = 0.05 * EDS_xSize
    for i in O.interactions:
        x1 = O.bodies[i.id1].state.pos[0]
        x2 = O.bodies[i.id2].state.pos[0]
        if((x1 < buffer_range) or (x2 < buffer_range)):
            i.phys.kn = 1000 * i.phys.kn
            i.phys.FnMax = 1000 * i.phys.FnMax
        elif((x1 > (1 - buffer_range)) or (x2 > (1 - buffer_range))):
            i.phys.kn = 1000 * i.phys.kn

```

```

i.phys.FnMax = 1000 * i.phys.FnMax
#E1,E2 = mat1.young, mat2.young
#i.phys.kn = i.phys.crossSection * 0.5*min(E1, E2) / i.phys.refLength

print("About to run! ")

O.run(table.max_iter)
O.wait()

print("done running! ")

#Count number of broken bonds between ITZ_clay
count_number_broken_ITZ = True
broken_ITZ_clay_count = 0
broken_ITZ_carbonate_count = 0
broken_ITZ_quartz_count = 0
broken_total_count = 0
if count_number_broken_ITZ:
    for i in O.interactions:
        if (i.phys.isCohesive == False):
            broken_total_count += 1
            mat1 = O.bodies[i.id1].mat
            mat2 = O.bodies[i.id2].mat
            if((mat1.label == 'ITZ_clay') or (mat2.label == 'ITZ_clay')):
                broken_ITZ_clay_count += 1
            elif((mat1.label == 'ITZ_carbonate') or (mat2.label == 'ITZ_carbonate')):
                broken_ITZ_carbonate_count += 1
            elif((mat1.label == 'ITZ_quartz') or (mat2.label == 'ITZ_quartz')):
                broken_ITZ_quartz_count += 1

print("Broken ITZ_clay count: " + str(broken_ITZ_clay_count))
print("Broken ITZ_carbonate count: " + str(broken_ITZ_carbonate_count))
print("Broken ITZ_quartz count: " + str(broken_ITZ_quartz_count))
print("Broken total count: " + str(broken_total_count))

##### printing out broken bonds information #####
fracname = "cracks_" + triax_cracks_filename + ".txt";
crack_matID_count = [0, 0, 0, 0, 0, 0, 0]
greater_than_EDSx_count = 0;
greater_than_EDSy_count = 0;
if (os.path.exists(fracname)):
    crack_array = [];
    cracks_x_array = [];
    cracks_y_array = [];
    break_num = 0;
    with open(fracname) as inputfile:
        csv_read_inputfile = csv.reader(inputfile, delimiter = ' ')

```

```

next(csv_read_inputfile)
for row in csv_read_inputfile:
    for i in range(1,4):
        xval = float(row[1])
        yval = float(row[2])
        normalized_xval = int((xval / true_EDS_scale));
        normalized_yval = int((yval / true_EDS_scale));
        if (normalized_xval > (EDS_xSize-1)):
            normalized_xval = (EDS_xSize-1)
            greater_than_EDSx_count += 1
        if (normalized_yval > (EDS_ySize-1)):
            normalized_yval = (EDS_ySize-1)
            greater_than_EDSy_count += 1
        matID = integrated_edges[normalized_yval][normalized_xval]
        crack_matID_count[sphere_mat_structure.matID_to_Ind_Dict[matID]]+=1
        crack_array.append([break_num, xval, yval, normalized_xval, normalized_yval, matID])
        cracks_x_array.append(xval / true_EDS_scale);
        cracks_y_array.append(yval / true_EDS_scale);
        break_num+=1
plt.tick_params(
    axis = 'both',
    which='both',
    bottom = 'off',
    top = 'off',
    labelbottom='off',
    labeltop = 'off')
imshow = plt.imshow(mat_map.final_map_with_interfaces, vmax=50)
plt.scatter( x=cracks_x_array, y=cracks_y_array, c='w',s=30)
plt.grid(b=None)
print("greater than EDS_x count: " + str(greater_than_EDSx_count))
print("greater than EDS_y count: " + str(greater_than_EDSy_count))
plt.savefig(cracks_filename_info)

```

#Output Saved Information In an External File

```

minStress = min(stressList);

def add_max_stress_to_summary_info(summary_filename, individual_filename, min_stress):
    column_headers = "Individual_filename;Minimum Stress;Number of Non-Cohesive Breaks;Number
of Broken ITZ_clay;kerogen count;clay count;carbonate count;quartz count;ITZ_clay
count;ITZ_carbonate count;ITZ_quartz count;ITZ_clay/total
count;kerogen_cracks_count;clay_cracks_count;carbonate_cracks_count;quartz_cracks_count;ITZ_clay_c
racks_count;ITZ_carbonate_cracks_count;ITZ_quartz_cracks_count"
    delimiter = ";";
    data_seq = [individual_filename, min_stress, broken_total_count, broken_ITZ_clay_count,
kerogen_count, clay_count, carbonate_count, quartz_count, ITZ_clay_count, ITZ_carbonate_count,
ITZ_quartz_count, 1.0*ITZ_clay_count/total_element]
    data_seq = data_seq+crack_matID_count
    for i in range(len(data_seq)): data_seq[i] = str(data_seq[i])
    append_line = delimiter.join(data_seq)
    ##### actually print data output

```

```

if (os.path.exists(summary_filename)):
    myfile = open(summary_filename, "a")
    myfile.write("\n")
    myfile.write(append_line)
    myfile.close()
else:
    myfile = open(summary_filename, "a")
    print("doesn't exist, creating a new file with filename = " + summary_filename)
    myfile.write("sep=;")
    myfile.write("\n")
    myfile.write(column_headers)
    myfile.write("\n")
    myfile.write(append_line)
    myfile.close()

    add_max_stress_to_summary_info(summary_filename, table.batch_single_filename_prefix + "_" +
str(table.temp_num), minStress)

print("\n-----FINISHED RUN-----\n")

```